KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Hardware-Backed Identity Management Systems

Jan VOSSAERT

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

September 2014

Hardware-Backed Identity Management Systems

Jan VOSSAERT

Jury:
Prof. dr. ir. Willy Sansen, chair
Prof. dr. ir. Bart De Decker, supervisor
Prof. dr. Vincent Naessens, supervisor
Prof. dr. ir. Bart Preneel
Prof. dr. ir. Lieven De Strycker
Prof. dr. ir. Christophe Huygens
Prof. dr. ing. Jeroen Boydens
Prof. dr. Marco Vieira
(University of Coimbra, Portugal)
Prof. dr. David Chadwick
(University of Kent, United Kingdom)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

September 2014

Uitgegeven in eigen beheer, Jan VOSSAERT, Celestijnenlaan 200A box 2402, B-3001 Heverlee (Belgium)

ISBN 978-94-6018-884-8

D/2014/7515/108

Acknowledgments

This text marks the conclusion of my doctoral work. Now, the time has finally come to thank all the people that have contributed to this work.

First of all I would like to thank my co-supervisor Prof. Vincent Naessens for delaying my entry in the industrial job market, introducing me to the academic community and giving me a chance to pursue this Ph.D. His valuable advice and encouragement guided me from a freshly graduated master student to the person hereby proudly presenting his thesis to you.

Secondly, I would like to thank Prof. Bart De Decker for being the supervisor of this thesis. His valuable feedback helped improve this work, every step along the way.

Many thanks further go to the members of the jury, Prof. Bart Preneel, Prof. Lieven De Strycker, Prof. Christophe Huygens, Prof. Jeroen Boydens, Prof. Marco Vieira and Prof. David Chadwick for accepting to be members of the jury and for their observations, which have improved the quality of this text. I want to thank Prof. Willy Sansen for chairing the jury.

I would like to thank my colleagues from the MSEC research group at KAHO Sint-Lieven, Jorn Lapon, Koen Decroix, Faysal Boukayoua and Laurens Lemaire. Having such a capable sounding board to bounce off ideas proved invaluable during my work on this thesis. Their contributions are, however, not only of professional nature. They also provided opportunities for distraction when required and ensured my time spent working on this thesis has been a very instructive experience, professionally as well as personally.

Many thanks further go to the many (former) colleagues from CODeS-KAHO: Wim, Jannes, Tony, Pieter, Joris x2, Greet, Jan, Túlio, Sam, Thomas, Eline and David. The same applies to the colleagues from DistriNet-Secanon: Kristof, Pieter, Milica, Andreas and Italo.

Last but not least I would like to thank my parents, sister and in particular

my girlfriend Fien. Although her academic contributions to this work are fairly small, her continuous support has been vital to the completion of this work.

Jan Vossaert
Ghent, September 2014

Abstract

This thesis explores how hardware security technologies can be applied to satisfy the diverse set of requirements inherent to identity management systems and technologies. This thesis focuses on the use of existing hardware-based security technologies by software rather than developing new hardware components. All concepts proposed in this thesis are subjected to an in-depth evaluation and are validated by means of a prototype.

The first part of this thesis deals with device authentication between resource-constrained nodes and more powerful devices. A security architecture is presented that allows low-cost resource-constrained devices to establish a secure authenticated channel with more powerful devices using symmetric-key cryptography. The constrained devices can enforce fine-grained access control policies based on the information obtained during authentication. In application domains with a limited scope a tamper-resistant module containing a common cryptographic key can be added to the constrained devices to increase the scalability of the key management process.

In the second part of this thesis a new user-centric identity management system is presented. Instead of relying on digital signature algorithms to assert the validity of attributes to service providers, the identity management system proposed in this part relies on a trusted application running on a tamperproof secure element. The service provider is assured that the received information originates from a genuine secure element in the system. The tamperproofness ensures that an attacker cannot directly access the memory of the secure element to extract or modify information. The identity management system combines several interesting features of existing governmental and federated identity management systems.

In the third part of this thesis a trusted execution environment that can be established on commodity workstations and laptops using TPM-based technologies is used to increase the security and privacy of existing identity

management technologies. Two complementary case studies are presented. The first case study focuses on the prevention of credential abuse through sharing or theft. The user's credential is, therefore, bound to one or more of his biometric traits. The second case study focuses on increasing the security and privacy of authentication infrastructures that rely on a smart card containing an X.509 credential and the identity information of the user. The system increases the security by allowing the user to enter his passcode via his workstation while protecting it from malware running on the operating system. The system increases the privacy of the user by giving the user more control over the disclosure of his information.

Beknopte samenvatting

Deze thesis onderzoekt hoe hardware-gebaseerde beveiligingstechnologieën kunnen ingezet worden om de diverse set aan vereisten inherent aan identity management systemen en technologieën te realiseren. Deze thesis focust op het gebruik van bestaande hardware-gebaseerde beveiligingstechnologieën via software in plaats van het ontwerpen van nieuwe hardware componenten. Alle concepten voorgesteld in deze thesis worden onderworpen aan een grondige evaluatie en worden gevalideerd aan de hand van een prototype.

Het eerste deel van deze thesis behandelt authenticatie tussen nodes met beperkte rekenkracht, geheugen en vermogen, zoals bijvoorbeeld draadloze sensoren, en krachtigere toestellen zoals een smartphone, laptop of werkstation. Er wordt een beveiligingsarchitectuur gebaseerd op symmetrische cryptografie voorgesteld. De beveiligingsarchitectuur laat toe om een veilig geauthentiseerd kanaal op te zetten tussen goedkope toestellen met beperkte middelen en krachtigere toestellen. In applicatiedomeinen met een beperkte scope kan een sabotagebestendige module met een gemeenschappelijke cryptografische sleutel toegevoegd worden om de schaalbaarheid van het sleutelbeheer te verbeteren.

In het tweede deel van deze thesis wordt een nieuw identity management systeem voorgesteld. In plaats van digitale handtekeningen te gebruiken om de geldigheid van gebruikersinformatie te attesteren aan een dienstverlener maakt het systeem gebruik van een vertrouwde applicatie die draait op een sabotagebestendige module. De dienstverlener wordt verzekerd dat de ontvangen informatie afkomstig is van een geldige module in het systeem. De sabotagebestendigheid zorgt er voor dat een aanvaller geen directe toegang heeft tot het geheugen van de module om zo informatie uit te lezen of te wijzigen. Het identity management systeem combineert verschillende interessante eigenschappen van bestaande identity management systemen.

TPM-gebaseerde technologieën maken het mogelijk om een vertrouwde omgeving te realiseren op werkstations en laptops. In het derde deel van deze

thesis wordt gebruik gemaakt van deze vertrouwde omgeving om de privacy- en veiligheidseigenschappen van bestaande identity management technologieën te verbeteren. Twee complementaire case studies worden voorgesteld. De eerste case studie focust op het voorkomen van misbruik via het delen of stelen van persoonlijke geheime sleutels. De geheime sleutels van de gebruikers worden hiervoor aan één of meerdere biometrische eigenschappen van de gebruiker gebonden. De tweede case studie focust op het verbeteren van de privacy- en veiligheidseigenschappen van authenticatiesystemen die gebruik maken van een smartcard met een X.509 credential, zoals bijvoorbeeld de Belgische elektronische identiteitskaart. Het systeem verhoogt de veiligheid door het ingeven van de pincode in het werkstation af te schermen van malware die draait in het besturingssysteem. Het systeem verhoogt de privacy door de gebruiker meer controle te geven over het vrijgeven van zijn informatie.

Abbreviations

A	Athlete
ACL	Access Control List
ANS	American National Standard
APDU	Application Protocol Data Unit
API	Application Programming Interface
AU	Audit Authority
B	Bracelet
BIOS	Basic Input Output System
C	Coach
CA	Certificate Authority
CCID	Circuit Card Interface Device
CI	Card Issuer
CRL	Certificate Revocation List
CRTM	Core Root of Trust for Measurement
CVC	Card Verifiable Certificate
DAA	Direct Anonymous Attestation
DMA	Direct Memory Access
DNT	Do Not Track
DRTM	Dynamic Root of Trust Measurement
ECC	Elliptic Curve Cryptography
ECC CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically Erasable Programmable Read-Only Memory

FIM	Federated Identity Management
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
IAIK	Institute for Applied Information Processing and Communication
IdM	Identity Management
IdP	Identity Provider
IWT	Agency for Innovation through Science and Technology
JCE	Java Cryptography Extension
L	Locker
LOA	Level Of Assurance
M	Middleware
MA	Management Authority
MAC	Message Authentication Code
MD	Mobile Device
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OS	Operating System
OTP	One-Time Password
P	Provider
PAL	Piece of Application Logic
PCR	Platform Configuration Register
PGP	Pretty Good Privacy
PIN	Personal Identification Number
PKI	Public-Key Infrastructure
QR Code	Quick Response Code
R	Reader
RA	(Re)Validation Authority
RFID	Radio-Frequency Identification
ROM	Read-Only Memory

RSA	Rivest Shamir Adleman
S	Smartphone
SAML	Security Assertion Markup Language
SE	Secure Element
SME	Small and Medium Enterprise
SoC	System on Chip
SP	Service Provider
SSL	Secure Sockets Layer
SSO	Single Sign-On
SVM	Secure Virtual Machine
TAN	Transaction Authentication Number
TCB	Trusted Computing Base
TCG	Trusted Computing Group
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TLV	Tag Length Value
TNC	Trusted Network Connect
TPM	Trusted Platform Module
TXT	Trusted Execution Technology
U	User
UHCI	Universal Host Controller Interface
UMA	User-Managed Access
USB	Universal Serial Bus
W	Workstation
WB	Web Browser
WS	Web Server
WSN	Wireless Sensor Node
XMHF	eXtensible and Modular Hypervisor Framework
XML	Extensible Markup Language

Contents

List of Figures

List of Tables

Chapter 1

Introduction

When the first commercial email system was introduced in the mid-seventies, it was one of the first analogue services being digitized. Today most analogue services have a digital counterpart. This varies from tax filing to shopping and social networks. Many of these services require the user to release personal information. In today's registration processes, service providers cannot rely on the correctness of user-disclosed attributes, opening the door for identity fraud. Many critical services require corroborated user information. Hence, simple password-based authentication is no longer sufficient to satisfy the security requirements of these services. This has led to the development of more sophisticated credential technologies such as X.509 and anonymous credentials [53, 58, 47]. These technologies allow the user to release reliable personal attributes during authentication, facilitating the development of online personalized services.

A credential technology is, however, only a small part in an entire Identity Management (IdM) system. An IdM system *supports administration of identity attributes including the development and choice of the partial identity and pseudonym to be (re-)used in a specific context or role [165].* An Identity Provider (IdP) is responsible for asserting one or more identity attributes of the user. Each identity provider can typically provide a specific set of attributes. For instance, the government can assert someone's name and address, while a university could assert someone's degree. Service providers can, subsequently, rely on these assertions to grant the user access to a specific service. For instance, students under the age twenty-five could be allowed to buy public transport tickets at a reduced rate. Some identity management systems assign Levels Of Assurance (LOAs) to each type of attribute provided

by an identity provider. Some identity providers are authoritative for a specific set of attributes while other identity providers may be able to provide the same attributes but with a lower level of assurance [60].

Several European countries are establishing a digital identity management infrastructure for their citizens by issuing electronic identity cards [153]. Contrary to their analogue predecessors, these electronic identity cards can also be used to authenticate in the digital domain. Governmental and commercial service providers can use this infrastructure to obtain reliable user information. The convenient and real-time nature of this process should further stimulate the transition from analogue to digital services. Although the government can assert many identity attributes, the presence of only one identity provider seriously limits the number and diversity of identity attributes supported by the system. Not only governments discovered the advantages of identity management infrastructures. Where companies used to have their own isolated authentication infrastructure, cooperation between companies spurred the development of Federated Identity Management (FIM) systems [60, 18]. These allow members of one organization to authenticate and access resources of another organization in the association. As a result, many identity management systems exist, each conceived within a specific context and each with its own advantages and constraints.

There are several hardware security technologies that can be used to realize the security and privacy requirements of these identity management systems. Secure Elements (SEs) such as smart cards have long been used to securely store the credentials of the user. However, the tamperproof properties of secure elements can have a broader use. Instead of relying on digital signatures to assert the authenticity of the user information the secure element can be used to assert the authenticity of the information. This principle is, for instance, used in the German electronic identity card [32] and in [35]. A second evolution is the use of the Trusted Platform Module (TPM) [186] and related technologies [68, 80] to realize a Trusted Execution Environment (TEE)[142] on commodity workstations. The TEE allows the execution of code in isolation from the operating system and, hence, from any malware running on the operating system. Embedding these technologies in identity management systems opens up many opportunities for research.

This thesis explores these two research directions and thereby tries to contribute to answering the question *how hardware based security technologies, more specifically secure element and trusted execution environment technologies, can be applied to realize the diverse set of requirements inherent to identity management systems and technologies.* This thesis focuses on the use of existing hardware-based security technologies by software rather than developing new hardware components.

1.1 Approach and Scope

Many identity management systems, both in academia and industry have already been presented. This PhD, however, addresses the issues related to identity management from a new perspective. We apply commodity hardware security technologies to increase the security, privacy and user-centricity of identity management systems. They can provide an alternative to conventional security technologies or they can be used to realize complementary security properties. No new cryptographic protocols or credential technologies are developed, instead this work strongly focuses on IdM architectures and on the innovative use of existing technologies. All concepts proposed in this PhD are subjected to an in-depth evaluation and are validated by means of a prototype. The focus of the prototypes is illustrating the feasibility of the system and not on providing a fully verified industry-ready implementation.

This approach is applied in three settings. In the first part, authentication between resource-constrained and more powerful devices is addressed. To satisfy these resource constraints, only lightweight cryptographic protocols are used. This work is performed on a two-year IWT[1] Technology Transfer (TeTra) project. There was close collaboration with local SMEs participating in the project to define the presented case studies. This type of projects aims at stimulating innovation by illustrating how existing technologies and knowledge can be used in concrete applications. In the second part, a new user-centric identity management architecture is developed. It combines concepts of different types of identity management systems to improve upon state of the art solutions. The last part focuses on using TPM technology to provide security and privacy enhancing solutions that can be plugged in to existing identity management systems. The research of the second and third part is conducted for an IWT Strategic Basic Research project.

1.1.1 Hardware-Backed Authentication using Symmetric Key Technology

The first part of this text deals with authentication between resource-constrained and more powerful devices. Many systems have already been developed with this concern in mind, each making different assumptions about the requirements of the resource-constrained devices. These devices can range from RFID tags to (wireless) sensor nodes. Even within RFID tags a categorization can be made in those that do not have cryptographic

[1]The IWT is the Flemish agency for innovation through science and technology and provides funding for innovative industry-relevant projects.

capabilities, tags that have basic capabilities such as symmetric encryption and hashing, up to tags that can even run asymmetric cryptographic algorithms. This text assumes a setting where resource-constrained nodes can execute basic cryptographic computations such as symmetric encryption and cryptographically secure pseudorandom number generator.

*As a **first main contribution**, a key infrastructure is presented that allows mutually authenticated key establishment between the resource-constrained and the powerful devices. Fine-grained access control policies can be deployed and enforced. The system is validated by applying it in several case studies. Its feasibility is illustrated by developing a prototype for a sports environment setting.*

The work presented in this part was first published [198] in the proceedings of the *Mobisec 2010* conference. An extended version [194] was published in a special issue of the *Security and Communication Networks* journal.

1.1.2 Secure Element-Backed Identity Management

The second part focuses on user authentication towards a service provider. Several systems have already been developed that address this use case. Federated identity management systems aim at increasing the user-friendliness of authentication procedures, while at the same time ensuring strong authentication to service providers. However, the user has typically limited or no control over the attributes that are exchanged between the identity provider and the service provider. Moreover, the identity provider can compile extensive profiles containing the services access by each user. Second, many countries are rolling out electronic identity technology. A majority uses a smart card that contains a few certificates and private keys that can be used to authenticate the user to multiple services. Service providers develop authentication and authorization modules that are eID-compliant. Although the user now has the credentials to authenticate to multiple services, current solutions pose many drawbacks. First, electronic identity cards typically only store static attributes (i.e. personal properties that do not change, such as name, date of birth, etc.). Moreover, users often have little impact on the attributes that are released during authentication. In some systems, they are always identifiable and they are required to release attributes that are not needed for the particular service.

The integration of anonymous credentials technologies in identity management systems can go a long way towards increasing the privacy of the user. They are, however, rarely used in practice. This is caused by several reasons: high complexity of cryptographic building blocks, performance demands,

compatibility with legacy infrastructure and complexity of the revocation procedure [128, 46].

*As the **second main contribution**, a new identity management architecture is presented. It uses technologies that are commonplace in today's identity management systems, such as X.509 credentials and smart cards while focusing on preserving the user's privacy. It combines several interesting features of existing governmental and federated identity management systems. Mechanisms for trust establishment between user, service- and identity providers are implemented. Special attention is paid to user control and policies. A prototype of the architecture is implemented and validated in multiple case studies.*

The identity management architecture was first published [195] in the proceedings of the *EuroPKI 2010* conference. An extended version [200] was published in a special issue of the *Mathematical and Computer Modelling* journal. The prototype was published [196] in the proceedings of the *PrimeLife/IFIP 2010 Summer School*. The validation case studies were published [42, 41] in the proceedings of the *PrimeLife/IFIP 2011 Summer School* and the *Mobisec 2012* conference.

1.1.3 TEE-Backed Identity Management

Policy makers are reluctant to replace existing infrastructures as this typically involves high costs. The third part of this research, therefore, takes a different approach. Instead of creating a new identity management architecture, a system complementary to existing architectures and technologies is developed to increase their security and privacy properties. To this end, TPM-based technologies are used to realize a trusted execution environment on the workstation of the user. The TEE can assure the user and the service provider that the authentication protocols are executed in a secure environment, in isolation from potential malware on the user's workstation. Two complementary case studies that illustrate the feasibility and added value of the approach are worked out in depth.

The first case study focuses on the prevention of credential abuse through sharing or theft. The most commonly applied solution is embedding the credential in a tamperproof module, such as a smart card. This impedes making digital copies of the credential. Users can, however, still pass their smart card to other users. Moreover, users sometimes pick an easy-to-guess passcode, which defeats the purpose of the passcode.

*The **third main contribution** presents a new solution for activating credentials. The credentials are bound to the biometrics of the user. This binding*

is verified in a TEE running on the workstation of the user. A prototype implementation of this system, requiring only commodity hardware, is presented to validate our solution.

This work was published [199] in the proceedings of the *CMS 2013* conference.

The second case study tackles several issues related to smart card-based electronic identity infrastructures. The user often enters his PIN via the workstation. This allows malware on the workstation to intercept it, which may lead to further abuse. Second, currently many systems use X.509 credentials for user authentication, which do not offer the same privacy-preserving properties as anonymous credential systems.

*As a **fourth main contribution**, a solution is presented that allows users to securely enter their PIN on their workstation to activate the credentials on their smart card. The solution further extends existing smart card assisted authentication technology based on X.509 credentials with privacy-preserving features such as multi-show unlinkability and selective disclosure. The system can, hence, be used to improve the privacy properties of these existing infrastructures. This is realized using a TEE environment on the workstation. A prototype implementation of this system validates our solution.*

This work was published [197] in the proceedings of the *SEC 2014* conference.

1.2 Outline

The rest of this dissertation is structured as follows:

Chapter 2: Identity Management Technologies shortly introduces the most important credential technologies and identity management systems. The strengths and weaknesses of these technologies and systems are discussed.

Part I: Hardware-Backed Authentication using Symmetric Key Technology

Chapter 3: Authenticated Key Establishment between Resource-Constrained nodes and Powerful Devices proposes a security architecture for authentication, access control and secure data transfer between resource-constrained devices and powerful stations.

Chapter 4: Lightweight Authentication: Validation and Case Studies validates the security architecture proposed in the previous chapter by applying it in multiple case studies. A prototype is implemented to demonstrate the feasibility of our approach.

Part II: SE-Backed Identity Management

Chapter 5: User-Centric Identity Management using Secure Elements presents a new approach for user-centric identity management that tackles several privacy and security problems in current systems.

Chapter 6: User-Centric Identity Management: Validation and Case Studies validates the identity management system proposed in the previous chapter by developing a prototype and applying it in multiple case studies.

Part III: TEE-Backed Identity Management

Chapter 6: An Architecture for TPM-Backed Identity Management introduces several concepts related to TPM-based TEEs and presents a security evaluation. This chapter additionally presents a solution that allows the user to establish trust in the software running on the TEE on the workstation, via his smartphone.

Chapter 8: TPM-Backed Identity Management: Validation and Case Studies uses the setup presented in the previous chapter to increase the security and privacy in existing identity management systems. Two complementary case studies are presented, each of which tackles distinct security and privacy issues related to identity management systems/technologies.

Chapter 9: General Conclusions.

Chapter 2

Identity Management Technologies

The main goal of this chapter is to establish the context in which the rest of this text can be viewed. The first section gives an overview of several credential technologies that are being used as building blocks for identity management systems. The second section discusses several identity management models and systems. The third section introduces the hardware-security technologies that are used in this work. Privacy-preserving identity management is only a small aspect in protecting the privacy of the user. The fourth section shortly introduces several additional technological challenges to protect the privacy of the user.

2.1 Preliminaries

2.1.1 Terminology

This section discusses the terminology used in the remainder of this text. The terminology presented by Pfitzmann et al. [165] is used to describe the privacy features of credentials technologies and identity management systems. An extensive survey on key establishment can be found in Chapter 12 of the *Handbook of Applied Cryptography* [144].

- A *pseudorandom function* is a function that can be used to generate

output from a random seed and a data variable such that the output is computationally indistinguishable from truly random output. For a formal definition of a pseudorandom function, please refer to [102]. In practice, pseudorandom functions can be implemented using, for instance, a block cipher.

- *Key establishment* is the process by which two entities establish a shared secret key. This key can subsequently be used to protect the confidentiality and integrity of data exchanged between both parties.
- *Implicit key authentication* is realized if both parties in the key agreement protocol are assured that no other entity aside from the identified other party can learn the value of the established secret key.
- An *authenticated key agreement protocol* is a key agreement protocol which provides implicit key authentication to both participating entities.
- *Key confirmation* is provided by a key agreement protocol if both parties in the key agreement protocol are assured that the other party actually has possession of the established secret key.
- *An authenticated key agreement with key confirmation* is a key agreement protocol which realizes both implicit key authentication and key confirmation to both participating entities.
- *Authentication encryption [31]* is encryption mode that simultaneously provides confidentiality, integrity and authenticity assurances on the data. It can either be realized using specific block cipher modes or by combining an encryption and MAC algorithm.

2.1.2 Notation

- $\mathsf{prf}_s(x)$ Denotes a pseudorandom function with seed s and input data x.
- $\{a\}_k$ Denotes the symmetric encryption of a with secret key k.
- $a||b$ Denotes the concatenation of binary strings a and b.

2.2 Credential Technologies

This section discusses three types of credentials that are commonly used in industrial or academic identity management systems namely password-based solutions, public key credentials and anonymous credentials.

2.2.1 Username and Password

A username/password combination is still the most commonly used type of credential for user authentication in a Web-based context. With the increasing amount of online services, users have to remember more and more passwords. User circumvent this problem by reusing passwords, by writing them down or by storing them in a file on the workstation. This, however, introduces several security and usability weaknesses.

Several systems have been developed to improve the security of password-based authentication. Two-factor authentication solutions [19, 17, 188] require users to prove knowledge of a valid username/password combination and possession of a second token. This token can, for instance, be a mobile phone [19] or a bookmark [17]. This prevents that a stolen username/password combination alone is sufficient to impersonate the user. Second, PwdHash [168] is a browser extension that transparently generates a different password for each service provider based on a master password entered by the user, some unique data associated with the service provider and a private salt stored on the client machine. The password is generated by applying a cryptographic hash on a combination of these sets of data. The user, hence, only needs to remember one password. To address the issue that passwords can be replayed, One-Time Passwords (OTPs) can be used. A one-time password is only valid for one login session. Additional technological support ensures that users do not need to retain a set of one-time passwords in memory. The most basic system prints a set of OTPs on a sheet of paper for each user. This is used in the Danish eID infrastructure [111]. Other systems, for instance, generate passwords based on the previous password, a challenge or the time. A well know OTP system is SecurID [169] from RSA Security. These OTP systems can also be used as a complementary element to username/password-based authentication to realize two-factor authentication.

Graphical passwords [67, 205, 36] is an approach to improve the usability of password-based authentication. During registration, the user clicks certain image areas or orders images in a certain order. This pattern is then used as a secret to authenticate the user. This approach assumes it is much easier for users to remember these patterns than a strong textual password. Each image can trigger the user to remember the unique secret pattern.

While several solutions have been proposed to increase the security and usability of password-based authentication, the main disadvantage remains that they do not allow the user to prove identity attributes. While this is an integral part of identity management systems, it does not mean that passwords are never used in these systems. They are typically used to activate another credential which

does allow proving identity attributes.

2.2.2 Public-Key Credentials

A public-key certificate [96] links the public key of the user to a set of information pertaining to the owner. During authentication, the user proves ownership of the certificate by proving possession of the corresponding private key. Multiple approaches can be used to establish trust [38] in the link between the public key and the user's information. The most common approach is working with a Certificate Authority (CA) that issues the certificates. The relying party trusts the CA and can verify the link between the public key and the user's information using the public key of the CA. Typically, a multi-layer architecture is used with multiple entities between the root CA and the end user's certificate. An alternative to the CA approach is working with a web of trust [15], as is done in PGP. In this approach, there is not one but multiple entities that endorse the link between the public key of the user and the identity, distributing trust over multiple entities. The most common type of public-key certificate is X.509, an ITU-T standard. The most broad scale adoption of X.509 credentials is the Public-Key Infrastructure (PKI) for server authentication in the HTTPS protocol.

Several electronic identity technologies use X.509 credentials for user authentication. During authentication all the information contained in the certificate is released to the service provider. This technology, hence, does not support selective disclosure where the user can select which information is released during authentication. This can be mitigated by linking identity files to the certificate instead of embedding all the identity information in the certificate itself. This allows the user to separate different types of information in different identity files. During authentication, only the identity file relevant to the specific service is released. For instance, online shops can request an address file but should not have access to the driver's license information of the user. An alternative approach is to include a cryptographic hash of the attribute values concatenated with a salt (i.e. $\text{Hash}(\text{value}_{at} \mid\mid \text{salt}_{at})$) in the credential [193], instead of the attribute value itself. This allows the user to selective disclose attributes by releasing the credential, the requested attribute values and the corresponding salts. The service provider can verify the integrity of the disclosed information but cannot derive the attribute values that are not released. However, since every authentication with the same credential is linkable, different service providers can merge their profiles of a user to compile one large profile. Hence, public-key credentials are mostly suited for server authentication and contexts where the credential is always used towards the same service provider.

Public-key credentials can be revoked based on a unique serial number contained in the credential. During revocation, the serial number is added to a Certificate Revocation List (CRL). During authentication, the relying party checks the revocation status of the credential by verifying that the serial number is not included in the CRL. Alternatively, the relying party can send the serial number to an Online Certificate Status Protocol (OCSP) server. The server checks the revocation status and sends a signed response back to the relying party.

2.2.3 Anonymous Credential Systems

Anonymous credential systems [65, 47, 53, 54] enable privacy preserving authentication. Where public key credentials force the user to release all information contained in the certificate, anonymous credential systems enable selective disclose of the information contained in the credential. The user can also choose to release no attribute information, only proving possession of a valid credential. Moreover, it also allows the user to prove unique service-specific pseudonyms, impeding efforts of service providers to cooperate and compile large user profiles.

All these features allow the user to minimize the information disclosed to service providers during authentication. These features, however, come at a certain cost. Anonymous credential systems are computationally much more expensive compared to regular public key credentials. This increases the duration of the authentication and the load on the server. Also the revocation protocols for anonymous credentials are much more complex than for public key credentials [128, 46]. The Certificate Revocation Lists (CRLs) used for public key credentials require a global identifier known to both the credential issuer and relying parties. This would defeat the unlinkability property achieved by anonymous credentials. Short-term credentials could be used as an alternative to revocation. This approach, however, also has several drawbacks. The computational load on the issuer increases since periodic credential updates are required. The issuer is required to be online to allow the user to periodically update his credential(s). Further, if a short-term credential is renewed on demand before it is used, the renewal and use of the credential may be linkable via timing correlation.

Anonymous credential systems are often split into two categories. Credential systems where the blinding is performed during the credential issuance phase (i.e. using blind signatures [64]) can only be used once if the user does not want to be linkable between transactions as the public key and certificate are revealed to the verifier. To prevent this linkability, a credential issuer can issue

several credentials to a user, who can dispose of a credential after using it. This technique is used in the U-Prove [47] credential system. For credentials where the blinding is executed during the credential show phase (i.e. using zero-knowledge proofs [103, 93, 73]), there is no limitation on the number of times it can be shown without degrading the privacy of the user. This system is applied in the Idemix framework [53, 54, 58]. The European ABC4Trust[1] project aims at defining a common unified architecture for these types of credential technologies and delivering an open reference implementation. In the IRMA[2] project, this type of credential technology is used to develop and evaluate an experimental framework for privacy-friendly attribute-based identity management using smart cards.

When users can access services anonymously/pseudonymously, they might be more inclined to try and abuse the service if they cannot be held accountable. Hence, measures need to be implemented to enable deanonymization if abuse is detected. Verifiable encryption [57] can be used to encrypt the identity of the user with the public key of a trusted third party. The service provider can verify that the correct identity is contained in the encryption without learning the actual value. The encryption can be randomized to ensure that multiple encryptions of the same identity cannot be linked. The trusted third party is trusted to only reveal the identity in case of abuse.

Despite the computationally intensive nature of anonymous credential systems several attempts have been made to port them to smart cards. Initially, only a limited set of the functionality was implemented [35, 178]. More recent implementations [150, 201] realize more of the functionality. Other approaches [34] execute only the security critical operations on the smart card and delegate the privacy-preserving calculations to the host, speeding-up the process. Although there are still some barriers for using anonymous credential systems (e.g. there is no universally good revocation strategy), these and other [29, 25] advances make anonymous credential systems an increasingly viable option for rolling out electronic identity infrastructures.

2.3 Identity Management Systems

This section discusses several identity management systems relevant to this work.

[1] ABC4Trust is an EU-funded research project, more information can be found on `http://abc4trust.eu`.

[2] IRMA is a project led by the Digital Security group of the Radboud University Nijmegen, more information can be found on `https://www.irmacard.org/`.

2.3.1 Web-Based Identity Management Systems

Several identity management systems such as Shibboleth [149], openID [167] and UMA [137] (User-Managed Access) have been developed to manage the identity of the user in a Web-based setting. Shibboleth [149] and openID [167] are two common examples of federated identity management systems [60, 18]. UMA is an identity management system developed by the Kantara Initiative[3] based on OAuth[4] for granting access to Web resources.

In federated identity management systems a user is known to at least one organization (i.e. the identity provider) in the federation (i.e. a group of organizations with mutual trust agreements). If a user contacts a service provider, authentication is delegated to the identity provider of the user. The user authenticates to the identity provider using one of the supported credential technologies. Subsequently, the identity provider releases the requested personal data to the service provider.

Shibboleth and OpenID can realize a high level of scalability and interoperability (e.g. based on standards and easy integration in Web services) but also have major drawbacks. For instance, a high level of trust is required in the identity providers. They are trusted to only release the required/requested attributes to the service provider and not to impersonate the user. Impersonation is possible since no authentication between the user and the service provider is required. The identity provider also knows which services a user is accessing. This allows the identity provider to protect the user's privacy by not transferring the user's personal information to blacklisted malicious service providers. It, however, also allows the identity provider to compile a user profile which consists of all the services accessed by a user. Moreover, combining attributes from different identity providers (i.e. attribute aggregation [61]) is not supported. The model is also not suited for offline use. Some initiatives try to mitigate some of these drawbacks. For instance, UApprove [159] is a Shibboleth extension that gives the user more control over the attributes exchanged between the identity and service provider. The identity provider is also a single point of failure since it actively participates in the authentication of its users.

In these types of systems, the service provider or a dedicated discovery service is trusted to redirect the user to the correct identity provider. A malicious service provider could redirect the user to a phishing server to obtain the login information of the user. Where Shibboleth is currently used in closed

[3]The Kantara Initiative is a non-profit professional association dedicated to advancing technical and legal innovation related to digital identity management. More information can be found on `http://kantarainitiative.org`.

[4]OAuth is an authorization framework for controlling access to a Web resource, more information can be found on `http://tools.ietf.org/html/rfc6749.html`.

federations, many OpenID providers have a much more open philosophy, aiming for the broadest possible adoption. In OpenID, each user has a global identifier (i.e. URL) that is released to the service provider. The latter can use the global identifier to locate the corresponding OpenID provider and handle the authentication procedure. OpenID systems are, therefore, more vulnerable to phishing attacks than Shibboleth. Further, several implementation flaws have been found with several popular OpenID providers [203] and TLS is not mandatory during the authentication process which an active attacker can abuse [187].

2.3.2 Governmental Identity Management systems

Several European countries are issuing electronic identity cards [153]. Many countries use a tamperproof device, typically a smart card, to store the authentication credential and personal information of the user.

- Many European countries such as Italy [22], Spain [110], Portugal [70] and Belgium [75, 76] use a smart card containing an RSA private key and corresponding X.509 certificate for user authentication. This credential is activated via the PIN of the user. The government provides a middleware package that needs to be installed on the workstation of the user and facilitates authentication of the user towards a remote service provider. The Belgian middleware, for instance, implements a `PKCS#11` module that other applications (e.g. the browser) can use to support authentication with the Belgian electronic identity infrastructure. It handles the communication to the card and allows the user to enter his PIN. This type of identity cards uses X.509 credentials without any privacy enhancing measures. They, hence, inherit many of the privacy issues related to that credential technology.
- In the German electronic identity card [32] system, a common key pair is shared by a large set of smart cards. This key pair is used to set up a secure authenticated channel between the card and the service provider. Since a key pair is shared with lots of other smart cards, no uniquely identifying information is released during the establishment of the secure channel. The number of smart cards with the same key pair should be high enough to provide an acceptable anonymity set. The card can now release user information over the secure channel. The service provider can only query a limited subset of the personal information stored on the card. This subset depends on the nature of the service and is regulated by the government. The policy is stored in the service provider's certificate, allowing the smart card to enforce it. Service provider-specific

pseudonyms are generated by the card, based on a card-specific secret and data in the service provider's certificate. The government knows the link between a user and the card-specific secret. This link is used to revoke identity cards. The revocation procedure consists of generating service provider-specific revocation lists containing the card-specific pseudonyms of each revoked card. Hence, when the number of service providers and revoked identity cards increases, the overhead of the revocation procedure increases considerably. Knowledge of the card-specific secret also allows the government to deanonymize the user based on a pseudonym (i.e. by generating the pseudonyms of all users for a specific service provider) or link profiles of users with different service providers. Relying on a key pair shared by a large set of smart cards introduces significant overhead if it is compromised. If a key pair is compromised, all smart cards containing that key pair need to be replaced. Since the system relies on the tamperproofness of the smart card to assert the validity of user attributes, compromising the tamperproofness allows an attacker to forge information.

- The Austrian [134] and Finnish [162] electronic identity card is a technology neutral concept that can be deployed on different valid physical tokens (i.e. devices that comply with Austrian/Finnish legislation for serving as an eID card). Moreover, multiple organizations can issue identity cards (e.g. banks issue ATM cards and one of the telecom operators issues SIM cards that are considered valid eID cards).

Governmental identity cards mitigate multiple drawbacks of federated identity management systems. The common identity domain model, however, also has several disadvantages. The model is not as flexible as federated IdM systems with respect to attribute provisioning. The model only supports one identity provider, limiting the diversity of attributes available in the system. Further, attributes are typically stored in the card during the whole card's lifetime. This implies that only (relatively) static attributes can be stored in the card. The procedure to modify the attributes is typically expensive (e.g. involving the user going to the town hall). To enrich the diversity of data available via the electronic identity card, the Belgian government developed a centralized platform that allows service providers to retrieve information about the user via a unique identifier obtained from the identity card. This centralized platform is connected to several remote databases that contain user information. These databases are managed by different entities and contain diverse sets of user information. When a request for user information is received, the centralized platform checks whether the requesting entity is allowed to access that information. If so, the requested information is retrieved from the databases and provided to the requesting entity. Gradually, more and more

information can be made available this way. In the eHealth domain, the social identity card is being phased out since the social security number and insurance status is now available through the centralized platform. This greatly increases the application domain of the identity card but at the same time increases the risks associated with the system. Although only certified instances can retrieve specific sets of information, the user has much less control over the exchange of his information since the whole process of retrieving the information via the central platform is transparent to the user. Abuse handling is mainly focused on logging, as each data request is logged in a database.

Further, since each country is rolling out its own electronic identity infrastructure, service providers would have to provide authentication modules specific for each country. To establish European electronic identity interoperability, the EU STORK[5] project adopted the federated identity management model. Each country provides an identity provider that asserts the identity of their citizens to (cross-border) service providers. The user uses his electronic identity card to authenticate towards the local identity provider. Service providers then only have to support the common authentication protocol adopted by all identity providers. This, however, also introduces some of the disadvantages of the federated identity management model such as increased trust in the identity providers.

2.3.3 Identity Management Systems Based on Anonymous Credentials

Several identity management systems have been proposed that use anonymous credential systems for user authentication [56, 132, 34, 182]. In [182], Suriadi et al. propose a user-centric federated single sign-on system based on the private credential system proposed by Bangerter et al. [27]. To alleviate the computational expensive nature of anonymous credential systems, the anonymous credential is only used to authenticate towards an identity provider, during which the required information is released. The identity provider provides SSO functionality to the user. It uses standards (i.e. Security Assertion Markup Language (SAML)[6] messages) to assert the identity of the user to service providers. Service providers are, hence, shielded from the complex nature of anonymous credential systems. In [34], the mobile device is used to manage the Idemix credentials of the user. Different identity providers can issue credentials containing information about the user. The mobile device scans a

[5]STORK is an EU-funded research project, more information can be found on `https://www.eid-stork.eu/`.

[6]SAML is an XML based standard for the exchange of authentication and authorization data.

QR code[7] containing the policy [55] of the service provider. Subsequently the mobile either generates a QR code containing a proof satisfying the service provider's policy or transfers that proof over the mobile's Internet connection to the service provider. While anonymous credential systems can be used to minimize the information disclosed during authentication, using those types of systems also presents some drawbacks. It increases the computational load on the server and client compared to X.509 credentials Further, supporting revocation is a challenging task.

2.4 Hardware Security Technologies

This text uses two types of hardware security technologies: trusted execution environment technologies and secure elements. Trusted execution environments technologies are contained in commodity hardware such as mobile phones or workstations. They provide a separate execution environment that protects the software running in that environment from malware that is potentially running on the operating system of the device. Where trusted execution environment technologies run on commodity hardware and mainly aim to protect against software attacks, secure elements are dedicated hardware components that run a trusted application and also provide protection against hardware attacks. Both types of technologies are discussed in more detail below.

2.4.1 Trusted Execution Environment Technologies

As defined by GolbalPlatform,[8] a Trusted Execution Environment [100] (TEE) is a separate execution environment that runs alongside a rich operating system (e.g. Android, Windows). The TEE provides security services for the rich environment and isolates access to its hardware and software security resources from the rich OS and its applications. This section shortly discusses two different technologies to realize a TEE. The ARM TrustZone technology is mainly used in mobile devices such as smartphones, while TCG Trusted Computing technologies are integrated in laptops and workstations.

[7] A Quick Response (QR) code is a two-dimensional machine-readable optical label.

[8] GlobalPlatform is a cross industry, non-profit association which identifies, develops and publishes specifications that promote the secure and interoperable deployment and management of multiple applications on secure chip technology - Wikipedia.

ARM TrustZone

ARM TrustZone [23] is a security extension embedded in several commercial ARM processors. An ARM physical CPU with TrustZone extensions is presented as two virtual CPUs. One virtual CPU runs a *normal world* rich environment such as Android and one virtual CPU executes *secure world* applications. The security extension supports restricting hardware resources (e.g. memory, screen, keyboard, notification LED) to applications running in the secure world. Applications running in the secure world have access to the full power of a device's main processor and memory, while applications running in the main operating system have restricted access.

Upon boot, the ARM processor starts executing in secure world mode. The software first loaded is, hence, secure world software. The secure world software is responsible for booting the rich operating system in the normal world. To establish trust in the secure world software, a secure boot procedure can be implemented. The secure boot procedure uses a chain of trust. The root of trust is the first software module that starts executing. Each software module is responsible for verifying the integrity of the software module(s) it loads. To enable integrity verification, each software module is signed by a trusted authority. This trusted authority is typically the platform vendor or a dedicated third party that manages the software running in the secure environment. In the context of the ARM TrustZone technology, especially the secure world software should be verified. To prevent simple hardware attacks, the root of trust should be located in the on-SoC ROM. On-SoC one-time-programmable hardware can be used to store (a cryptographic hash of) the public key used by the root of trust to verify the software module(s) it loads.

An important remark to make is that ARM's hardware platform architecture is only a specification. Device manufactures are free to customize a specific implementation. The potential security features offered by the ARM TrustZone technology are, hence, not always realized in practice [191].

Giesecke & Devrient developed a secure world middleware layer and application framework called MobiCore [98] to facilitate the development and deployment of applications running in the ARM secure world. So far Giesecke & Devrient's MobiCore was integrated in the Samsung Galaxy S3 and the Galaxy Note II. Sierraware implemented the SierraTEE and SierraVisor TEE solution [99], which is freely available under the GNU GPL v2 license for the Samsung Exynos 4412 and nVIDIA Tegra 3 SoCs.

TCG Trusted Computing

TCG Trusted Computing relies on a separate hardware module (i.e. a Trusted Platform Module [186] (TPM)) that is physically attached to the computer's motherboard, extending the system with a set of security related features. Most commodity laptops and workstations are nowadays equipped with a TPM. It can be used to measure and attest the software stack running on the system. The TPM can also be used to seal data to a specified trusted software stack. The sealed data can then only be accessed if the device is running the software stack specified during sealing.

The trusted execution technologies [68, 80] embedded in several chipsets from Intel and AMD allow the execution of measured code independently of previously executed software. It can, hence, be used to interleave the regular operating system with the execution of TEE applications. Both the regular OS and the TEE applications are in full control of the hardware. Before executing the TEE software, the processor measures the code via the TPM (i.e. a cryptographic hash of the TEE software binary is stored in the TPM). Hardware protections such as disabling interrupts, preventing DMA access to the code in memory and even prohibiting access by hardware debuggers attached to the motherboard, ensure that the measured code is the code executed by the processor. The secure measurement of the code by the TPM ensures that data can be locked to the application and is, hence, not available to any other application running on the system.

Evaluation

There are several important distinctions between the ARM TrustZone and the TCG Trusted Computing approach to realize a trusted execution environment. First, with ARM TrustZone technology each device attached to the bus can, if it supports the required bus protocols, distinguish between secure and normal world requests. This allows hardware resources to be restricted to the secure world. In the TCG approach, both the applications running in the trusted execution environment and the rich operating system are in full control of the hardware. The only device that can distinguish between TEE applications and normal applications is the TPM. The TPM is used to seal the data of the TEE applications so that it cannot be accessed by any other software component running on the system. It is, hence, much easier to inform the user whether or not a trusted application is running in a TrustZone system (e.g. by using a LED only accessible from the secure world) compared to a TCG-based system. Knowing when a trusted TEE application is running is a key factor in establishing a trusted path [209] between the user and a hardware resource

such as (a smart card. Second, the mechanism to establish trust in the TEE software differs between both approaches. In ARM TrustZone, a secure boot mechanism is used to ensure that the software running in the secure world is trustworthy. In the TCG approach, the software attestation capabilities of the TPM are used to establish trust in the TEE applications. Software attestation is a mechanism that allows asserting which software components are running on a workstation to a local or remote party. This makes the TCG approach more open compared to the TrustZone system. To develop and deploy applications for a TrustZone-based trusted execution environment on end-user devices such as smartphones, cooperation of hardware integrators such as Samsung or framework (e.g. MobiCore) providers such as Giesecke & Devrient is required.

A trusted execution environment can also be implemented through pure software virtualization (e.g. Xen, OKL4 Microvisor). These solutions, however, lack the additional security through hardware support. The hardware security extensions, for instance, provide protections against DMA attacks that can be used to add malicious software to memory.

2.4.2 Secure Element

As defined by GlobalPlatform, a secure element is a tamper resistant platform (typically a one chip secure microcontroller) capable of securely hosting applications and their confidential and cryptographic data in accordance with the rules and security requirements set forth by a set of well-identified trusted authorities. These authorities are trusted by the different actors interacting with the secure element (e.g. user, service provider). Secure elements are most commonly deployed in the form of smart cards. They are typically used in applications with strong security requirements such as payment applications, electronic identity systems and access control systems.

A secure element is, typically, a resource constrained device. Most secure elements do not allow direct interaction with a user but have to rely on a dedicated reader or a general purpose device such as a workstation or laptop to interact with the user. This can introduce security risks since the software running on these devices is not as trustworthy as the software running on the secure element. This can make the system vulnerable to malware attacks [101, 26, 120, 175]. Malware could, for instance, intercept the passcode of the user or mislead the user about the data that is actually signed by the secure element. A trusted execution environment could be used to tackle these challenges. The TEE could, for instance, offer a trustworthy user interface to

securely transmit a passcode to the secure element and inform the user about the transaction before it is processed by the secure element.

2.5 Supporting Privacy-Preserving Authentication

This text follows the *privacy as control* paradigm as defined by Guerses et al. [107]. It focuses on systems that increase the control users have over the disclosure of their information. Several identity management systems try to increase the privacy of the user by implementing mechanisms to give the user more control over the disclosure of his information and minimize the data released during authentication. However, service providers or third parties have many other techniques at their disposal to gather (privacy sensitive) information about the user. Users are, typically, much less aware about which information is gathered using these techniques and have, hence, virtually no control over this type of information gathering. This section gives a short overview of several techniques service providers or third parties can use to gather information about the user and what techniques can be used to limit the impact.

Users often access remote services via their browser. Web technologies allow many ways for service providers to gather information about users. Cookies have been developed with the explicit goal to recognize a user over multiple page requests. The goal was to improve user experience and allow stateful services by providing dynamic user-specific content. It, however, does not stop there. Since many websites import third-party content in their sites (e.g. for advertising, statistics, content-delivery), these third parties can use cookies to track a user over the Internet. This not only provides them with tracking data, it also allows them to gather general information about the user [127]. Some efforts try to raise awareness about this large scale tracking on the Web. For instance, Lightbeam [6] is a Firefox add-on that visually shows which third parties are contacted when you visit a specific website. In the Netherlands, service providers are obliged to inform the user if they use cookies. As awareness rises, users might start deleting their cookies more often. Hence, some service providers employed additional technologies such as HTML5 and flash storage to store uniquely identifying information on the computer of the user, creating so called Supercookies that are very hard to remove from the browser. Other initiatives try to block tracking. For instance, NoScript [10] and Ghostery [3] are browser add-ons that block requests to tracking sites. Recently, most browsers added support for the Do Not Track (DNT) header. This is a HTTP header field that requests Web applications to disable its tracking of an individual user. The header is, however, not legally

binding, leaving service providers free to either honor or completely ignore the request. A recent study [16] showed that this header is completely disregarded. Moreover, apart from cookies, the browser itself can also be used to identify the user [87, 16]. Service providers can also retrieve other information about the user. For instance, the Geolocation API in HTML5 enables service providers, given permission of the user, to obtain the location of the user via the browser.

The nature of the provided service also allows the service provider to obtain information about the user. Mail providers can scan the content of mails for targeted advertising and online shops offer personalized recommendations based on the shopping behavior of the user. Some service providers also learn information about users from what other users release. For instance, users upload photos of other users on social networks and tag users in photos, events, locations etc.

With the increasing popularity of smartphones, more and more services are being accessed via the smartphone of the user. To be able to provide more user-friendly and rich services, service providers are switching from Web services to developing dedicated applications. These applications often require numerous permissions [112], allowing them to access the rich sources of information (e.g. contacts, location, SMSes and calendar data) on the mobile device. Some of these permissions are necessary due to the nature of the application, but often applications require permissions that are not strictly necessary for the application to function correctly [112]. Moreover, users are typically unaware of how the information is used and whether or not it is transferred to remote parties. Hence, for meaningful deployment of privacy-preserving identity management systems on mobile devices, the user needs more control over the access and use of data by mobile applications. This is the subject of much current research [91, 33, 112, 152, 171, 24].

The communication layer also allows third parties to gather information about the user. The IP address alone can be used to identify a user and learn its location. Traffic analysis can be used to compile large profiles of the browsing behavior of the user. Anonymous communication protocols [183, 81, 147, 151, 146, 88] have been developed to impede efforts to gather information of the user based on network communication.

2.6 Conclusions

This section gave an overview of credential technologies and identity management models. Each technology and model has their own (dis)advantages.

Common identity domain systems are well suited for authentication in a specific domain. The limited diversity of available information and the limited scope of a domain with only a specific set of service providers mitigates the privacy implications of public-key credentials. However, to increase the application domain, electronic identity infrastructures are being extended to provide an increasingly diverse set of information. Moreover, this information is no longer stored on the identity card of the user but available to service providers with virtually no control of the user. Common identity domain systems using public-key cryptography without any additional privacy-preserving measures are insufficient to protect the privacy of the user in this setting. The federated SSO identity model provides a scalable architecture for exchanging information between providers. The control over the exchanged information, however, lies largely with the identity provider. Moreover, the system is not usable in offline scenarios and most implementations allow the identity provider to profile the user regarding the visited services and do not support attribute aggregation. User-centric identity management tackles most of these shortcomings. These solutions, however, typically rely on anonymous credential technologies which are computationally more expensive and increase the management burden and are, therefore, hardly used in practice.

Currently most identity management systems rely on the workstation to provide a secure environment for handling the user's credentials. Although identity management systems can incorporate tamperproof modules, they are often activated and accessed via the workstation. This allows malware potentially running on the workstation to intercept and abuse the activation credentials. Most systems deal with this threat in a reactive way, by revoking credentials. It would be better to handle this proactively and try to prevent this abuse from occurring.

Chapter 3

Authenticated Key Establishment between Resource-Constrained Nodes and Powerful Devices

Today, PKI solutions are widely adopted to establish secure authenticated communication channels between devices. Standards such as SSL/TLS were developed to facilitate the establishment of these secure channels. The HTTPS communication protocol, for instance, uses these protocols to set up a secure communication channel between a client workstation and a Web server. Public-key infrastructures allow tackling the security requirements that apply in open environments (i.e. secure communication between parties that have no previous relation) between computationally powerful devices. Symmetric cryptography is, however, superior in terms of performance, energy consumption and cost. Therefore, symmetric key solutions are often adopted in settings where these parameters are of paramount importance. Typical examples are small devices such as sensors that need to be online for long periods without an external power source. These sensors can, for instance, be used in body area networks or wireless sensor networks that sense environmental parameters (e.g. such as oscillators in volcano areas or temperature sensors in transport monitoring systems).

Many protocols [173, 63, 86, 92, 136, 45] have been developed to secure communication in a network of (lightweight) devices using symmetric key

solutions. However, those networks are not isolated. Today, many scenarios exist in which (networks of) lightweight devices need to release data securely and implement strict access control policies towards more powerful devices. For instance, body area networks can be used in health monitoring systems. The sensors may release parameters and status information to mobiles or gateways for statistical analysis or to detect and start alert procedures. Caregivers can read specific parameters with their smartphones to assess the situation of the patient. In supply chain systems, several environmental parameters need to be monitored and be available for inspection by authorities. These networks of constrained devices can be managed by an administrator who issues rights to other parties to activate and use certain functionality of the constrained nodes. The data that may be retrieved often depends on certain roles or specific privileges. For instance, a wireless node that is installed in a truck or attached to a container on a ship may contain a lot of information. A border control entity can read out a detailed history (e.g. route, driving and rest times) while relay stations or warehouses can only read out product related information (such as temperature history). Outsiders are not allowed to read out any information or eavesdrop on data that is exchanged.

These use cases require reader devices from multiple organizations, and hence with tailored privileges, to access specific services from constrained nodes. This chapter presents a security architecture for authentication, access control and secure data transfer between resource constrained devices and powerful stations, often with networking capabilities, that tackles these requirements. Symmetric key operations are performed on both sides. Asymmetric operations might be required by the powerful device when acquiring the necessary privileges from a management service. The systems supports commodity hardware such as smartphones to be used as reader devices. Our approach is compared to several other solutions.

3.1 Related Work

Radio-Frequency Identification (RFID) tags are small, wireless devices to identify objects and people. They are proliferating in trillions [121] due to the dropping costs. Juels [121] gives an overview of existing RFID tag designs and their applications, and evaluates their security and privacy properties. They are classified into two categories: *basic RFID tags* and *symmetric-key tags*. The former cannot perform cryptographic operations. Although the lack of cryptography is a big impediment to the security design, a few lightweight technical approaches can address certain security and privacy concerns. Some basic tags can be killed, put to sleep or blocked. Also, certain tags are protected

with a passcode. However, passcodes can be eavesdropped on. Symmetric-key tags have richer cryptographic capabilities which are typically used for (mutual) authentication, and for the prevention of tracking/tracing and cloning attacks. Multiple security designs are presented in the literature [181, 158, 148]. However, these solutions mostly focus on identification. More complex scenarios with multiple actors with tailored privileges and advanced access control requirements regarding the data stored on the tags are not supported. High-end tags which are, for instance, used in e-passports [122] also support public-key cryptographic operations [138, 28, 130, 204, 113, 131]. Most RFID security research focuses mostly on identifying the tag and authenticating data to (trusted) readers. This work focuses more on nodes that maintain and/or generate a richer set of information or provide personalized services. Hence, the constrained node needs to be able to enforce fine grained access control policies and a secure channel needs to be established between the constrained nodes and the reader to securely transfer data.

Wireless smart cards can also be used for radio-frequency identification. For instance, MIFARE technology is often used in access control application (e.g. public transport, tolling). In these settings, the card is merely a mobile storage medium for data that can be read and updated by dedicated readers in the system. The entire infrastructure (i.e. constrained nodes and readers) is typically managed by a single authority (e.g. the public transport company). There also exist high-end tamperproof smart cards that can have richer cryptographic capabilities such as RSA and ECC. These cards can, typically, run customs application allowing them to provide tailored support for the requirement of each specific use cases. They are, however, more expensive and only support very close range communication (i.e. around ten centimeters). They are mostly used in high security applications such as electronic payment and identity management.

Many lightweight cryptographic libraries [116, 59] containing (optimized) versions of symmetric cryptographic algorithms are available. They provide implementations of lightweight cryptographic blocks (e.g. GRAIN [166], PRESENT [40] and Trivium [74]) or optimized implementations of traditional cryptographic blocks (e.g. DES and AES). For instance, the IAIK cryptographic libraries [115] of the Graz University of Technology contain several implementations of the AES algorithm. They are often optimized for specific processor types, and designed for minimal energy consumption and high speed encryption.

Many solutions have been proposed to secure and authenticate traffic within wireless sensor networks using symmetric cryptography [173, 63, 86, 92, 136]. Contrary to these proposals, our work focuses on authentication of external devices to sensor nodes. Multiple entities may require access to specific sets of

data or services from sensors. Each entity can have different privileges, hence, fine-grained access control policies should be supported. Several systems [202, 174, 50, 109, 108, 210] have been proposed that rely on public-key cryptography to control access from external readers to wireless sensor networks. This chapter investigates how similar requirements can be tackled under the assumption that constrained nodes can only use symmetric cryptography and compares the proposed solution to systems using public-key cryptography.

3.2 Design

This section presents a security architecture that tackles the requirements that apply in settings where readers, both general purpose readers such as smartphones or dedicated readers, need to mutually authenticate with resource constrained nodes. This section first lists the different roles in the system, after which the requirements and adversary model are discussed. This section ends with the presentation of our approach.

3.2.1 Roles

The system administrator manages a set of resource constrained devices (cd). Third parties can request access to certain services or data contained on the constrained nodes using a powerful reader device (pd).

3.2.2 Requirements and Adversary Model

Resource Requirements

This chapter poses some general resource requirements that aim at minimizing energy consumption and cost while achieving reasonable performance in settings where powerful devices from a potentially diverse set of owners (and corresponding privileges) need to establish secure authenticated sessions with constrained devices.

R_1 *Resource constrained devices* only execute symmetric key operations for reasons of performance and cost. These nodes might be offline. Even though ECC algorithms have been highly optimized [28, 130, 204], symmetric algorithms [89] are still significantly more efficient. The protocols should have a low communication overhead (i.e. limited

data transfer and short distance communication protocols) as wireless communication can have an even bigger impact on the autonomy of devices than the cryptographic operations [160].

Security Requirements

S_1 The system should support authenticated key agreement between resource constrained and powerful nodes so that data can securely be exchanged between both parties.

S_2 The constrained nodes carry or generate sensitive data. Not all entities should be able to access all the data and services that are available from a node. The system should, therefore, allow fine-grained access control policies to be enforced.

Adversary Model

With respect to the cryptographic capabilities of the attacker, we follow the Dolev-Yao attacker model [85]: attackers cannot break cryptographic primitives, but they can perform protocol-level attacks. The administrator is trusted to issue capabilities to readers reflecting the privileges of their owner and to correctly deploy and configure the constrained nodes. This entails enforcing the desired access control policies and running software that correctly realizes the specified functionality.

3.2.3 Approach

This section presents the approach taken to satisfy the requirements presented in the previous section. First, the key infrastructure is presented. The second section illustrates how the key infrastructure can be used to establish a secure authenticated channel between a resource constrained and powerful device. The last section discusses revocation. In the protocol listings, authenticated encryption is assumed. Hence, apart from the confidentiality, also the integrity and authenticity of encrypted messages is protected.

Key Infrastructure

The system administrator is responsible for issuing and initializing the resource constrained devices with a symmetric key (K_M). Different settings can have

different requirements regarding key distribution. For instance, each device can store a unique K_M, K_M can be shared by a small group of devices or by all devices managed by the administrator. In Chapter 4 our approach is applied in two distinct settings. Each constrained device is also equipped with a dedicated secret key K_{cd} and seed for pseudo-random number generation. The administrator maintains a mapping between each K_M and a serial number. To enable authentication between the constrained nodes and powerful devices, the system administrator issues *capabilities* to powerful devices. A *capability (CAP_X)* contains the identity and rights of its owner and a random value. A secret key K_X and the serial number assigned to the K_M used to construct the capability are also issued to the owner. K_X and CAP_X are defined as follows.

- $K_X := \mathsf{prf}_{K_M}(rand)$
- $CAP_X := \{id_X||role_X||rand\}_{K_M}$.

Mutually Authenticated Key Agreement

During the key agreement protocol, illustrated in Table 3.1, the powerful device *pd* generates a challenge and sends it together with its capability CAP_{pd} to the constrained device *cd* (1-2). The latter decrypts CAP_{pd}, generates a challenge and calculates K_{pd} and K_s (3-5). The challenge is then sent to *pd*. The latter can now calculate the session key K_s using K_{pd} (6-7). An additional round can be added to realize key confirmation.

Table 3.1: Key agreement protocol between a constrained (cd) and powerful (pd) device.

		keyAgreement():
(1)	pd :	$chal_1 := \mathsf{genChallenge}()$
(2)	$cd \leftarrow pd$:	CAP_{pd}, $chal_1$
(3)	cd :	$[id_{pd}\|\|role_{pd}\|\|rand] := \mathsf{symDecrypt}(CAP_{pd}, K_M)$
(4)	cd :	$chal_2 := \mathsf{prf}_{K_{cd}}(seed_{cd}{+}{+})$
(5)	cd :	$K_{pd} := \mathsf{prf}_{K_M}(rand)$; $K_s := \mathsf{prf}_{K_{pd}}(chal_1 \mid\mid chal_2)$
(6)	$cd \rightarrow pd$:	$chal_2$
(7)	pd :	$K_s := \mathsf{prf}_{K_{pd}}(chal_1 \mid\mid chal_2)$

The challenges generated by both parties ensure the freshness of the session key and key control by both parties. Authentication and identification are realized through key confirmation. If key confirmation is completed, *cd* has proven knowledge of K_M and is, hence, authenticated (i.e. a genuine constrained

device managed by the administrator). The constrained node is also uniquely identifiable if each constrained device is equipped with a unique K_M. This does require the reader to have a separate CAP_X and K_X for each constrained device it interacts with.

Further, *cd* knows that the other party possesses the secret K_{pd} and, hence, knows that the data contained in the *capability* (i.e. *id*, *role* and *rand*) is associated with that party. Note that the protocol does not offer forward secrecy, since the session key can be reconstructed if either K_M or K_{pd} is compromised.

Revocation

To mitigate the impact of compromised capabilities several approaches can be taken, some of which can be combined to improve efficiency. The best approach depends on the setting in which the systems is used. In Chapter 4 our approach is applied in two distinct settings, in each of which a different approach is used.

- A serial number could be included in the capability, allowing the constrained nodes to check whether a capability is compromised by matching the serial number with a list of revoked serial numbers. This approach requires the constrained nodes to have sufficient storage to store the list of revoked serial numbers and to be able to contact a revocation service at regular intervals to update the list. The required storage space can be limited using certificate revocation trees [125] based on Merkle's hash trees. To further limit the required storage, each constrained node could be equipped with an additional secret key that is shared with the administrator. The administrator can then periodically update K_M using that additional secret key. This effectively invalidates all previously issued capabilities. Since the constrained devices typically only have short range communication capabilities, this approach scales badly in most settings.
- The contents of the capability can also be extended with a validity interval. If short-lived capabilities are used, the vulnerability interval of compromised capabilities is short. To check the validity interval, the constrained device is, however, required to have a real-time clock. In [157] a solution is proposed for constrained devices without real-time clock. The constrained node interacts with its owner to verify the validity period. This approach can, however, only be used in settings where each constrained device is under constant supervision by a user trusted to correctly verify the validity period.

When a K_M needs to be revoked, the administrator adds the serial number assigned to that K_M to a revocation list. Before using a capability, the powerful device checks the revocation status of the K_M used to construct the capability by verifying that the serial number received during the issuance of the capability is not included in the revocation list.

3.3 Evaluation

3.3.1 Security Evaluation

This section briefly discusses how the protocol protects against several well known attacks.

- **Replay attack.** Replay attacks are prevented since both parties generate a challenge based on which the session key is generated. Hence, old messages cannot be reused to authenticate successfully.
- **Man-in-the-middle attack.** Except for the trusted administrator, K_M is only known by the constrained node and K_X is only known by the owner of the corresponding CAP_X and generated by the constrained node during authentication. During authentication only CAP_X and two random challenges are sent in the clear. The session key is generated using a keyed (K_X) cryptographically secure pseudo-random number generator using the two challenges as a seed. Hence, an attacker requires K_X to generate the session key. To obtain K_X from the exchanged messages an attacker would need K_M. Hence, only the constrained node and the powerful device can generate the session key K_s, ensuring a secure end-to-end channel. In settings where K_M is shared by a group of constrained nodes, the constrained node could execute a man-in-the-middle attack. These devices are, however, managed and controlled by the system administrator.
- **Denial-of-service attack.** Since, message authentication is a cryptographic operation, it is possible for a malicious device to trigger expensive operations on the constrained nodes. An attacker could, for instance, send random data to the node which would trigger an integrity check. Timeouts after a number of failed authentication attempts could mitigate this problem.
- **Mafia fraud attack.** A mafia fraud attack is a type of relay attack in which the attacker forwards messages between an honest prover and

an honest verifier. The approach does not foresee a specific strategy to prevent this type of attacks. To prevent such attacks several approaches can be taken. A dedicated mechanisms to activate the authentication functionality on the constrained node could be used. Straightforward solutions use a slider or button; in [72] a more complex approach is proposed in which a dedicated activation pattern is measured with an accelerometer. Additionally, after the key agreement, a distance bounding [44] phase could be added to ensure that the reader and the constrained node are in each other's vicinity. The efficiency and effectiveness of many current distance bounding protocols is, however, limited [69, 30].

- **Hardware/malware attacks.** Hardware or malware attacks can be used to obtain K_M or the credentials of the powerful devices. The revocation strategies presented in the previous section can be used to limit the impact.

3.3.2 Requirements Review

The key agreement protocol requires only one pass (i.e. message exchange to and from each party) for the two parties to generate the session key. The exchanged data consists of two challenges and one encrypted value containing three entries of limited length. The required cryptographic operations can be realized using a symmetric cipher. Either lightweight implementations [115] of traditional ciphers such as AES or implementations of lightweight ciphers such as PRESENT [40] can be used to realize the cryptographic protocols (cf. R_1). The latter has superior performance when implemented in hardware but decreased security parameters with an 80-bit key and a 64-bit block size. An overview of the performance results of lightweight cipher implementations in hardware and software is given in [89]. In [106], several authenticated encryption modes using the AES algorithm were implemented and optimized for a specific microcontroller family.

As illustrated by the security evaluation above, the key agreement protocol realizes strong mutual authentication and provides protection against several well-known attacks. All sensitive information is sent over the secure channel established during authentication (cf. S_1).

The constrained node can enforce fine grained access control policies based on information that is embedded in the capability of the powerful device (i.e. a unique identifier and role). The *capabilities* can also be extended to support application-specific needs. The policy enforcement component restricts the information released over the secure channel (cf. S_2). This approach is well

suited in settings where capabilities have a long lifetime and where the policies on the constrained nodes are easily updated. In settings where capabilities have a short lifetime, however, it is better to have the system administrator embed the actual allowed actions in the capability instead of the role. The access control policy is, hence, specified centrally with the system administrator. This is illustrated in the following chapter, where the approach is applied in two distinct settings.

3.4 Comparison with Existing Systems

Several other protocols for authentication based on symmetric cryptography have been developed. For instance, the 3GPP AKA [211] system is used for authentication between mobile phones and the telecommunication network. Kerberos [145] is used in enterprise networks to control access of workstation to networked services such as a file server. MIFARE technology is often used in access control applications. Compared to approaches using public-key cryptography, the main downsides of the approach proposed in this chapter are that it does not realize forward secrecy and key management is more complex. The upside is the increased performance for lower cost of the constrained nodes resulting from the use of symmetric cryptography.

In Kerberos, all devices share a long term secret with a key distribution center. This key allows the key distribution center to issue tickets that allow two devices to set up a secure mutually authenticated channel. Contrary to our approach, Kerberos mostly focuses on authentication between devices managed by the same domain. Hence, the access control decision is taken centrally by the administrator. The system does not support enforcing fine grained access control policies locally. Revocation is also checked centrally, hence, tokens are always issued on-the-fly and cannot be used offline for prolonged periods.

In the 3GPP AKA scheme, each SIM card (and hence each mobile phone) shares a secret key with the telecom operator that issued the respective SIM. Base stations can retrieve tokens from an authentication center to mutually authenticate towards a mobile device. There is no need and, hence, no support for the SIM to differentiate between different base station. Our system focuses on settings where multiple readers have different roles and, hence, different access rights to information stored on the constrained node.

On a MIFARE card, the data is split into different blocks on the card. Each block can either be read, modified or is inaccessible by a reader, depending on the keys held by the reader. The keys to authenticate towards the cards are shared between readers with the same privileges. The card can be used

in multiple application domains by storing the data of each application in a separate block. The access control policy is determined during system deployment, making it ideal for static access control application such as public transport and tolling. Our system focuses on systems where policies are modified during runtime.

3.5 Conclusion

This chapter presented a security architecture allowing authenticated key establishment between resource constrained and powerful nodes. The system assumes short range communication capabilities of the constrained node and requires the constrained node to only execute symmetric cryptographic operations ensuring good performance and efficiency. Having only short range communication capabilities impedes updating revocation lists on constrained nodes. Hence, an approach with short-lived capabilities is the most scalable technique to limit the impact of compromised capabilities in most settings. The constrained nodes are managed by an administrator. The administrator also issues credentials to the different actors in the system, allowing them to authenticate and establish a secure channel with the constrained nodes. The constrained nodes can enforce access control policies based on the information contained in the credentials of the actors. An informal security analysis of the protocol is presented. The following chapter applies this security architecture to tackle the requirements in two different types of systems. For each type of system, a concrete case study is worked out.

Chapter 4

Lightweight Authentication: Validation and Case Studies

Electronic devices are more than ever integrated in our everyday environment. Some of these devices are powerful such as workstation, laptops or smartphones, others are more resource constrained. For instance, small sensors can be used in supply chain systems and patient monitoring systems [172, 43, 184, 20]. Some health monitoring sensors are not only used by patients, also athletes use heart rate sensors to achieve optimal training results. Powerful devices such as a smartphone or base station can gather the data generated by the sensors. Wireless implantable medical devices [77, 104] such as pacemakers not only monitor but also safeguard the health of the user. These implantable devices typically need to be initialized with patient specific parameters. Only the physician of the user should be able to read and change these parameters.

Some devices provide a direct interface to the user, others require an additional device such as a smartphone. These constrained devices might contain sensitive data or provide security critical services (e.g. implantable medical devices). Hence, a security infrastructure needs to be deployed that allows these powerful devices to authenticate and establish a secure channel towards the constrained nodes. In Chapter 3, a security architecture was presented that specifically aims at tackling the security and resource requirements of those settings. It requires the constrained nodes to only execute symmetric key operations, minimizing the cost and the power consumption. The system allows the resource constrained nodes to enforce fine grained access control policies. The proposed architecture is generic and can be applied in many settings that require powerful/constrained device interaction. This chapter illustrates how

the security architecture can be applied to tackle the requirements in both open and closed environments. For each environment a case study is presented. The case study of the closed environment is worked out in detail and a prototype is implemented. Both case studies are developed during a Technology Transfer (TeTra) project, in close collaboration with local SMEs.

4.1 Closed Environments

In closed environments where the scope is limited to one organization, the number of constrained nodes and powerful devices is limited. All devices are certified by a single administrative entity. Every constrained device can be equipped with a common K_M that is hardware protected. It can, for instance, be stored in on-chip ROM of the microcontroller; a more high-cost solution would be to use tamperproof modules. The limited scope of the closed environment should ensure that the cost for extracting the key is higher than the potential gains for the attacker. Since a common K_M is used for all constrained nodes, each powerful device can be equipped with a single capability allowing them to access services and data from constrained nodes based on their privileges. The constrained device proves to be genuine during the authentication. To allow the powerful device to identify the constrained node, the on-chip ROM can also store other information, such as a unique identifier which can be sent over the secure channel established during authentication. This additional information and K_M never leave the controller unencrypted, ensuring that it is not possible for an attacker to retrieve or modify this data (information and K_M) without resorting to invasive or side-channel attacks.

Several use cases can be developed in closed environments. For instance, smart homes may contain several sensors and controllers that can be configured or read using the smartphone or workstation of the user. Finally, in a sports club, each athlete can receive a bracelet that is used to access/consume multiple services of that club. This last case study is discussed in more detail in the remainder of this section.

4.1.1 Sports Association

Many sports associations already have their own website on which information is (publicly) available. Examples of available information are training and contest schedules, results and events. Some websites also provide personalized Web pages. Sports clubs also provide several other services. Examples

are entrance control, access to lockers, a free refreshment after a match, etc. Currently, each type of service requires their own type of credentials. Password-based authentication is typically used to control access to the website, analogue keys are used to access lockers and printed tickets are used to pay for consumptions. These services could be made more user-friendly, more reliable and more secure by digitizing them.

The case study discussed in this section presents an attractive solution to integrate multiple electronic services in the context of a sports association. The solution is tailored to youngsters and makes use of a wireless chip embedded in a bracelet. It realizes a reasonable trade-off between multiple (often conflicting) requirements. First, the bracelet must be low-cost and low-power. On the other hand, security and privacy are major concerns. For instance, locker and refreshments services involve important security requirements. Since sensitive personal data (such as contact information) is kept in the bracelet, privacy is also a crucial concern. The security architecture presented in Chapter 3 is applied to tackle these requirements.

Roles

The sports club has a *Web Server (WS)* providing a centralized platform to manage and disseminate information of the club. An *athlete* (A) is a member of a sports club. After registration, each athlete receives a *bracelet (B)*. This bracelets grants access to *Lockers (L)* and the online platform, via a *Workstation (W)*, of the sports club. A *coach* (C) is responsible for a group of athletes (typically with the same age, grade, or level). An administrator initializes and issues the bracelets to athletes. The administrator also issues capabilities to coaches to read and update information on the bracelets using his *Smartphone (S)*. The administrator is responsible for maintaining the IT infrastructure.

The athlete can benefit from multiple services using this bracelet:

- *Personalized Web service.* Athletes can log in on the website of the sports association using the security tokens stored in the bracelet. The personalized Web pages provide detailed information about training and contest schedules (e.g. time and type), personal achievements, contest results, etc. Moreover, the athlete can submit personal information (personal physician, phone numbers, whereabouts, ...). Also, after logging in, the user can update information in the bracelet (personal achievements, phone numbers, ...), modify the access control policies and download tokens to access lockers and to pay for drinks at the

sports canteen. The coach can update the content of the website using a dedicated application on his mobile phone.

- *Information retrieval service.* For instance, some athletes are willing to disclose/compare personal achievements with their friends. Further, phone numbers of relatives and/or physicians can be read by coaches in case of injuries, especially for very young athletes.
- *Locker service.* Each locker is equipped with a small embedded device with wireless communication capabilities that controls the opening and closing of the locker. Registered athletes have access to one or more free lockers via their bracelet. The maximum number of lockers that can be used simultaneously can depend on several parameters (e.g. registration fee, sport's type or age of the athlete). The coach can also open the locker of an athlete. This might be necessary in case the athlete lost his bracelet. Support for accountability should prevent the coach from abusing this privilege.
- *Refreshments service.* Each bracelet can contain a number of free or prepaid tickets for refreshments in the sport club's canteen. Those tokens can be downloaded from the website into the bracelet at regular times.

Security and Privacy Requirements

This section describes the most important non-functional requirements.

- *Access control.* Athletes must be able to control access to the data stored in the bracelet. Only a subset of information is released when the bracelet interacts with other components. The data that are released depend on the type of information and the privacy preferences of the user. The athlete's personal pages on the website of the sports club can only be accessed by the athlete itself. Lockers closed by an athlete can only be opened by the same athlete or one of the coaches. In the latter case, the system could provide some accountability (e.g. by logging this fact).
- *Location privacy.* It must not be possible to track athletes based on information obtained over the application layer. Tracking based on, for instance, the unique radio frequency pattern of the antenna is out of scope of this text.
- *Reliability.* Appropriate back-up and revocation mechanisms must be implemented (in case of lost or stolen bracelets and mobiles).

- *Confidentiality.* Transmitted data cannot be eavesdropped by external attackers.

Security Infrastructure

Figure 4.1 gives an overview of the devices in the ecosystem and the data stored on each device. This is further discussed below.

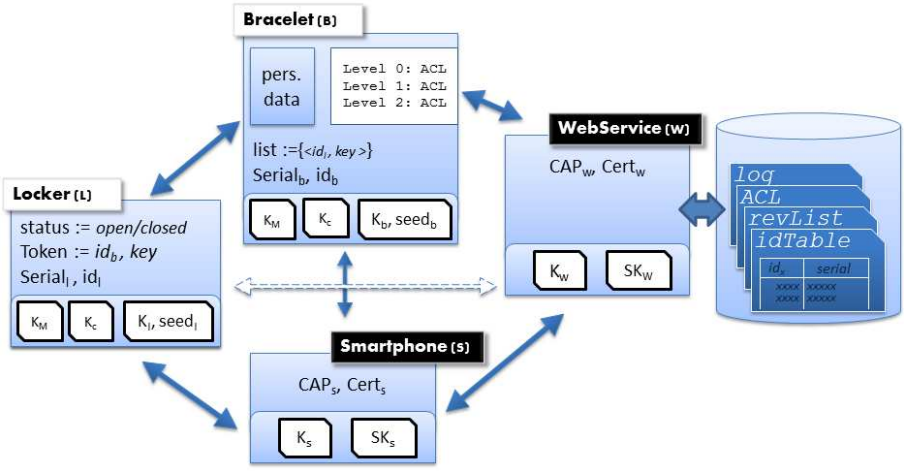

Figure 4.1: Overview of information storage, key infrastructure and capabilities.

Constrained Devices. Each sports association owns a set of lockers and bracelets. They should be low cost and low power. They do have NFC capabilities to interact with other devices. For reasons of performance, public-key cryptography cannot be performed by the bracelets and lockers.

The administrator manages the symmetric secret K_M that is stored on each constrained device during initialization. During initialization, the administrator also assigns an identifier (i.e. id_b/id_l), newly generated serial number (i.e. $serial_b/serial_l$) and a secret key (i.e. K_b/K_l) to generate cryptographically secure pseudo-random numbers. The lockers and bracelets also store a separate secret key (K_c) to secure communication between lockers and bracelets. The bracelet contains a tamper-resistant module on which this information is stored. The lockers are embedded in a physically locked secure

casing, preventing access to anyone but the administrator. This allows the administrator to update the keys if they are compromised.

Each bracelet also stores an Access Control List (ACL) and a set of personal data. The ACL and personal data are initialized by the administrator and can later partially be modified/updated by the athlete using the Web service. The bracelet also keeps a list containing the identifiers id_l of lockers closed by the bracelet and the tokens to open them. A green/red light shortly lights up when a protocol has been successful. Note that each bracelet also contains a button to activate it for a short period. This protects against relay attacks or malicious readers trying to connect without knowledge of the athlete.

Powerful Devices. The sports club manages a Web server running a website and content management software. Each coach has a smartphone with wireless communication capabilities to read out the information stored in the bracelet and update the information contained on the website. The Web server and smartphones both hold a public-key credential (i.e. respectively $cert_w/sk_w$ and $cert_s/sk_s$) for authentication between powerful devices via standardized protocols and a capability, CAP_X, and corresponding K_X for authentication with bracelets and lockers. The Web server also maintains a revocation list for both constrained and powerful devices. Athletes can log in with the bracelet on the Web server via a workstation or laptop. These devices should, hence, be able to connect with the bracelet. Some laptops have NFC (e.g. to interact with wireless smart cards) built-in. For workstations, on which these wireless communication capabilities are rarely available by default, an external reader can be used.

Revocation. The capability of the Web server grants the necessary rights to update a revocation list on the bracelet. This allows revocation of capabilities issued to coaches or other potential parties in the system based on the serial number contained in the capability. If the key of the Web server is compromised, new bracelets need to be issued and the keys on the lockers need to be updated.

The access of individual bracelets to the Web server can be revoked by adding the serial number of the revoked bracelet to a revocation list. If the shared K_M or K_c is compromised, new bracelets are rolled-out, the capabilities of the different actors are updated and the lockers are updated with new keys.

Protocols

This section presents the protocols to realize the services proposed in the beginning of this section. The refreshment service is not discussed in detail as the protocol is similar to that of the information retrieval service. Also, the use case of the coach updating information on the website using a dedicated smartphone application is not further discussed since it can be realized with standardized authentication algorithms using public-key cryptography.

Personalized Web Service. To access a personalized Web page (see also Table 4.1), the athlete activates the bracelet (1) and browses to the right *url* on his workstation (2). Next, an HTTPS session is set up: the Web server authenticates to the browser using $cert_W$ (3). The Web server and bracelet then agree on a mutually authenticated session key (4). The browser is extended to forward communication between both parties. Next, the bracelet encrypts his ID id_b and serial number $serial_b$ with the session key (K_s) and sends the result to the Web server where it is decrypted (5). If $serial_b$ is still valid, id_b is used to retrieve the corresponding identity from the database and a customized page is sent to the client browser (6-8). The athlete can now view personalized information (such as training schedules) and update some data (such as contact information). The Web server also allows the athlete to securely update information stored in the bracelet using K_s. The Web server can now also update the list of revoked capabilities on the bracelet.

Table 4.1: Retrieving personalized website after mutual authentication (A: Athlete, W: Workstation, B: Bracelet, WS: Web Server).

accessPersonalizedWebPage():

(1)	A :	activate_bracelet()
(2)	$A \rightarrow W \rightarrow WS$:	requestService(url, `"personalized_website"`)
(3)	$W \leftrightarrows WS$:	setUpHTTPS($cert_w$, sk_w)
(4)	$B \leftrightarrows W \leftrightarrows WS$:	(K_s; id_w; $role_w$) := keyAgreement(K_M, CAP_w, K_w)
(5)	$B \rightarrow W \rightarrow WS$:	$\{id_b, serial_b\}_{K_s}$
(6)	WS :	**if** (isValid($serial_b$) == **false**) abort()
(7)	WS :	$personal\_page$:= genPersonalPage(id_b)
(8)	$W \leftarrow WS$:	$personal\_page$

Controlled Release of Data. An athlete can specify its own privacy policy. Three access control levels are defined in the policy. Level 0 defines which information can be read by any device. No security tokens are required to access the information. Level 1 defines role-based access control rules. Level 2

defines information that should only be released to certain individuals (i.e. more restrictive than roles).

Table 4.2 shows the protocol for releasing privacy-sensitive data. First, the bracelet is activated, after which the coach instructs his smartphone to read out data from the bracelet (1-2). The mobile and bracelet then agree on an mutually authenticated session key (3). Next, the bracelet selects the data (i.e. $data_b$) that can be released to id_X with $role_X$ (4). The bracelet encrypts the concatenation of $data_b$ and its serial number $serial_b$ with K_s, and sends the result to S (5). The smartphone decrypts the data and checks the revocation status of the bracelet. The smartphone either periodically downloads a revocation list from the Web server or performs an online revocation check. Subsequently, the data is displayed to the coach (6).

Table 4.2: Releasing personal information to coach (A: Athlete, C: Coach, S: Smartphone, B: Bracelet).

releasePersonalInformation():

(1)	A	:	activate_bracelet()
(2)	$C \rightarrow S$	:	request($read\_data$)
(3)	$S \leftrightarrows B$	:	$(K_s; id_s; role_s) :=$ keyAgreement(K_M, CAP_s, K_s)
(4)	B	:	$data_b :=$ privilegedData(id_C, $role_C$)
(5)	$S \leftarrow B$	:	$\{data_b \| \| serial_b\}_{K_s}$
(6)	$C \leftarrow S$	:	$data_b$

Locker Service. The bracelet contains a number `MAX` that defines the maximum number of lockers that the owner of a bracelet may close simultaneously, and also a counter `nb_closed` that counts how many lockers are currently in use by the bracelet's owner. Moreover, the bracelet keeps a list of the identifiers (id_l) of the lockers in use and the corresponding tokens key to reopen them.

Closing a Locker. This protocol is shown in Table 4.3. The athlete activates the bracelet by pressing. Once activated, the bracelet can be detected by the locker which is periodically scanning for nearby NFC devices. If a nearby bracelet is detected, the locker checks its status (i.e. opened or closed). If the locker is open, it initiates the *close* protocol (1-2). The locker generates a key to open the locker after the *close* protocol is complete and a session key (K_s) (3). The locker encrypts the session key with K_c. Subsequently, the concatenation of key and the identifier of the locker are encrypted with the session key. These encrypted values are sent to the bracelet that holds K_c and can, hence, obtain the decrypted values (4). The bracelet now verifies that it can still close a locker (i.e. $nb\_closed < \texttt{MAX}$) and, subsequently, stores the identifier of the locker together with the key to reopen it (5-6). The bracelet checks if the

identifier of the locker is not already in the list of closed lockers. If it is in the list, the key is updated and *nb_closed* is not incremented. This ensures that failed runs of the protocol do not block the system (e.g. if the bracelet is removed from the proximity of the locker before the protocol is completed). The bracelet now encrypts its identifier and serial number with the session key and sends the resulting value to the locker where it is decrypted (7). The locker verifies the revocation status of the bracelet and subsequently stores the identifier of the bracelet with the *key* to open it and updates its status (8-10). The protocol for opening a locker is similar to the close protocol and is, hence, omitted.

Recovery Procedure. A mechanism is implemented to ensure that lockers can be opened when bracelets are lost. A trivial solution uses a physical key. The key can be used to open the locker. After opening, the embedded device can be reset. A more advanced solution gives privileges to (certain) coaches (and/or administrators) to open lockers. The coach can set up an NFC connection between his mobile device and the locker. To open the locker, the coach then authenticates with his smartphone to the Web server (e.g. over the mobile Internet connection of the smartphone). The smartphone then submits a request to open a locker to the Web service. The Web server verifies the identity and role of the requesting entity. Next, a secure connection is established between the locker and the Web server. The mobile phone of the coach is used to forward data between the locker and the Web server and the *capability* CAP_w of the Web sever is used to establish the secure authenticated channel. The Web server then sends an open request to the locker. After the privileges are checked, the locker sends the identifier of the bracelet to the Web server and then opens. Finally, the Web server logs the transaction (i.e. it stores the identifier of the coach, the locker and the athlete that closed the locker). Athletes will be notified that one of their lockers was opened when logging in to the Web service.

Prototype

A prototype was implemented based on the presented protocols. Table 4.4 shows the hardware used for the realization of the different entities. For the functionality of the bracelet, a Java Card [5] applet is developed and deployed on a wireless smart card. The NFC capabilities of the smartphone are used to interact with the smart card. An Android application is written that uses its capability to authenticate towards the smart card, read the information and visualize it to the user. The Web services is implemented on an Apache Tomcat [1] server. A Java Applet was developed and integrated in the website to bootstrap communication between the back-end service and the smart

Table 4.3: Closing a locker with a bracelet (B: Bracelet, A: Athlete, L: Locker).

closeLocker():

(1)	$B \leftarrow A$:	activate_bracelet()
(2)	L :	detectDevice() ($status$ == "open")
(3)	L :	key, K_s := $\mathsf{prf}_{K_l}(counter_l{+}{+})$
(4)	$B \leftarrow L$:	close($\{K_s\}_{K_c}$, $\{key \,||id_l\}_{K_s}$)
(5)	B :	**if** (numberOfLockersClosed() == MAX) abort()
(6)	B :	**if** (store(id_l, key) == false) $nb\_closed{+}{+}$
(7)	$B \rightarrow L$:	$\{id_b \| \| serial_b\}_{K_s}$
(8)	L :	**if** (isValid($serial_b$) == **false**) abort()
(9)	L :	store(id_b, key); $status$:= "closed"
(10)	L :	close()

card, via a wireless smart card reader. The back-end service and smart card execute the authentication protocol after which the service marks the session as authenticated and grants the appropriate rights. The locker was implemented by the DraMCo research group, the electronics partner in the TeTra project. The random number generation and encryption were realized using 128-bit AES. A cryptographic Message Authentication Code (MAC) was generated based on the ciphertext to realize authenticated encryption. Since the CMAC algorithm is not available on the smart card, HMAC [126] with RIPEMD-160 [82] is used to generate the MAC.

Table 4.4: The hardware platforms for the realization of the different entities (W: Workstation, WS: Web Server).

Entity	**Realization**
Bracelet	Gemalto TOP DM GX4 smart card
W, WS	DELL E4200: Intel Core2 Duo U9400 @ 1.4GHz, 4GB RAM The laptop runs Ubuntu 12.04 32-bit, 3.2.0 kernel
Locker	Atmel ATMEGA128 controller with NXP PN532 NFC-SoC
Smartphone	Nexus S: 1GHz ARM Cortex-A8, 512MB RAM The smartphone runs Android 4.1.2

Evaluation

A similar approach can be used in several other settings. For instance, a fitness center can give visitors access to their infrastructure for a limited period of time (linear to the amount of money that is paid). A bracelet can keep credits to consume drinks, avoiding the need to have cash at hand in the fitness

center. The credits can be downloaded to the bracelet after payment using the web service. The payment desk or automatic dispenser can decrease the credit. Multiple strategies exist to implement credits. Either a counter can be increased/decreased or signed tokens can be downloaded. The former strategy is more flexible whereas the latter is more secure.

The security architecture presented in Chapter 3 is used to establish secure authenticated channels between constrained nodes and powerful devices. Since K_M is shared by all constrained nodes, the powerful devices only need one capability, ensuring scalability and supporting offline authentication. Revocation of bracelets and powerful devices is supported by means of revocation lists. The bracelets are periodically updated with a new revocation list via the website. The limited scope ensures that the revocation list remains manageable and that it is, with respect to security, justifiable to use a K_M shared between all constrained devices. Athletes can control access to the information stored on their bracelets by specifying access control policies and loading them on to the bracelet via the website of the sports club. When closing a locker, the bracelet stores a token that is required to open the locker and prevents other athletes from opening his locker. When a coach opens a locker of an athlete, for instance in case of a lost or broken bracelet, the transaction is logged by the back-end and the athlete is informed via the website ensuring accountability in case of abuse. The locker protocol relies on the close range nature of the used communication protocols to thwart possible attacks in which a second bracelet stores the token to open the locker while the athlete is under the impression it is stored on his bracelet. The light embedded in the bracelet also provides an indicator that the protocol was completed successfully. The activation button attached to the bracelet ensures that it cannot be used for tracking.

4.2 Open Environments

In an open environment, multiple different actors, each possibly acting under a different authority with tailored privileges, require access to services or data from one or more constrained devices. Resource constrained devices are further referred to as *nodes*. These constrained nodes are managed by a node manager who is responsible for their deployment and maintenance. Each actor may require access to different services and data from the constrained node. Open environments typically contain a large number of constrained nodes. Storing a shared K_M in on-chip ROM would provide insufficient protection in these environments and equipping each constrained node with a tamperproof module

would come at a very high cost. Hence, each (group of) constrained node(s) is equipped with a unique K_M.

Equipping each reader with a separate capability for each node is not scalable. Hence, capabilities are issued on demand by the node's manager. The manager maintains a database containing the K_M stored on each node under his authority. When data or services are requested from a particular node, the reader first authenticates to the manager (e.g. an online Web service). An X.509 credential can be used in this phase. The manager assigns a set of rights based on the attributes in the certificate of the requesting entity and embeds them in a capability to access a particular node (with a unique K_M). The capability and corresponding secret key are then returned. To construct the right capability, the identity of the node must be known by the node manager. A unique identifier can be transmitted by the node to the reader. The latter can forward it to the node manager.

Typical open systems use cases can be found in the domain of supply chain management. Multiple parameters of items or goods in the supply chain are monitored using nodes in warehouses, sensors on trucks and on the supplies. This case study is discussed in more detail in the remainder of this section. Other potential domains are RFID ecosystems where the subset of information that can be read and/or updated depends on specific privileges. In [95] a detailed overview of potential applications of RFID in healthcare scenarios is given.

4.2.1 Supply Chain Management

This section applies the security architecture presented in Chapter 3 to tackle the security requirements of a supply chain management case study. The validation is scenario-based, no implementation was made.

Wireless sensors in the supply chain

Cargo containers can carry *wireless sensor nodes (WSN)* that form a *wireless sensor network*. These sensors gather (environmental) data, control actuators such as air conditioning systems and/or store information about the content of the containers. In harbours, *unloading companies* may want to learn the location of individual parcels. *Distribution centers* can request a detailed environmental history. Moreover, *quality control agencies* need physical and logical access to containers to verify relevant parameters of transport conditions. The latter can also store quality control information on the wireless nodes.

The *owner* of the goods can define and update thresholds for environmental conditions (such as temperature) that need to be satisfied during transport. *Forensic investigations* may need to retrieve detailed information from the sensors (e.g. who had access to the container and who updated certain parameters). Further, *customs* require support from the sensor network to obtain relevant information about the supply chain. For instance, they want to know which products are stored in each container, their origin, their route etc. All sensors/constrained nodes in a specific container could be equipped with the same K_M. This allows the entities in the supply chain to prefetch the necessary capabilities each day based on the the work schedule. This ensures that the reader devices do not constantly need to contact the manager.

Each sensor network is maintained by a management authority *MA* that initializes the sensors with the required keys and issues capabilities to the actors who want to interact with their sensors. Each actor can have a mobile device (e.g. a dedicated reader or a smartphone) for that purpose. For instance, *quality control agencies* and *police forces* could have a smartphone that enables them to perform the required operations while *unloading companies* could have a dedicated reader integrated in their unloading equipment. It is not scalable for the node manager to regularly update revocation lists on the constrained nodes, especially since separate revocation lists need to be maintained for each K_M. Hence, capabilities with small validity intervals are used to limit the vulnerability interval of compromised K_Xs. The sensors are, hence, assumed to have a real-time clock. In supply chain systems the validity interval could be limited to a day since containers in transit typically won't have interactions with the same entity for prolonged periods. Since the capabilities have a short lifetime and it is, typically, not easy to push policy updates to the constrained nodes, the node manager embeds the actual privileges of the powerful devices in the capabilities instead of the roles as proposed in the previous chapter. This decreases the complexity and increases the manageability of the system.

Protocols

To retrieve data from a wireless sensor node (see also Table 4.5), the reader (R) submits an identification request to that node (1). The latter returns a unique identifier *sensorID*. The reader forwards the *sensorID* to the node manager. It, therefore, first initiates a mutually authenticated TLS connection with the management service (3). All further communication between the reader and the management service is transmitted over the secure channel. After authentication, the reader transmits the *sensorID* to the back-end system that retrieves the node's key from the database (4–5). Next, a capability that contains the identity information and validity period is generated and

transmitted to the device (6–7). The reader performs the authenticated key agreement protocol with the node using the obtained capability and transmits a request for service X to the node (8–9). The node verifies whether the reader has the privilege corresponding to the requested service, after which the data is released or the service is executed (10–11).

Table 4.5: Requesting services from the wireless sensor network (WSN: Wireless Sensor Node, R: Reader, MA: Management Authority).

requestService():

(1)	$WSN \leftarrow R$		:	`"service_request"`
(2)	$WSN \rightarrow R$		:	$sensorID$
(3)		$R \leftrightarrows MA$	:	setUpHTTPS($cert_m$,sk_m, $cert_r$, sk_r)
(4)		$R \rightarrow MA$	:	$sensorID$
(5)		MA	:	K_M := database.getKey($sensorID$)
(6)		MA	:	CAP_r, K_r := genCred($cert_r$, rand, valPeriod, K_M)
(7)		$R \leftarrow MA$	:	CAP_r, K_r
(8)	$WSN \leftrightarrows R$		:	K_s := keyAgreement(K_M, CAP_r, K_r)
(9)	$WSN \leftarrow R$		:	$\{$`service_X_request`$\}_{K_s}$
(10)	WSN		:	**if**(verifyPrivilege(CAP_X, X) == **false**) abort()
(11)	$WSN \leftrightarrows R$		:	offerService(X)

Evaluation

Since capabilities are issued on demand by the node manager service, the revocation status of the node can be verified centrally using a revocation list. Short-lived capabilities are used to mitigate the impact of compromised capabilities. The main disadvantage of applying the presented approach in an open environment is that the node manager and reader devices are required to be online to issue/receive capabilities. Readers can prefetch capabilities to support authentication in offline environments. The validation is purely scenario-based. This evaluation is, hence, constrained by absence of a prototype.

4.3 Conclusion

This chapter applied the security architecture presented in Chapter 3 for authentication between resource constrained and powerful devices to tackle the requirements of both open and closed environment. For each type, a concrete case study is discussed, validating our approach. In closed environments, a

key can be shared between all constrained devices. This greatly facilitates key management. The limited scope of the system justifies the risk of the shared key set up in these types of systems. In open systems, an online service needs to be available that provides on-demand capabilities to readers. One case study is worked out in depth illustrating the feasibility of our approach.

Chapter 5

User-Centric Identity Management Using Secure Elements

In user-centric identity management systems the user is in control over his identity information flowing between service and identity providers. Hence, no identity information is exchanged directly between service and identity providers. To further increase the control and privacy of the user, features such as selective disclosure in which the user can select which identity attributes are released can be added. Currently most systems adhering to these principals rely on anonymous credentials. This significantly impacts the performance of these systems due to the computational load on client and server during authentication and the complex revocation strategies. In current anonymous credential systems, the performance, amongst others, depends on the number of attributes contained in the credential. Hence, increasing the number of attributes supported by the IdM system further degrades the performance. This chapter, therefore, focuses on an alternative approach for user-centric identity management relying on public-key credentials and tamperproof hardware. It supports attribute aggregation [62]: attributes can be retrieved from multiple identity providers and released to a service provider. It tackles several privacy and security problems of current federated identity management systems. Identity providers can no longer impersonate users and do not learn which services a user is accessing. Similarly, service providers are not necessarily aware with which identity provider the user is registered. In some cases, this might be relevant information that also needs to be disclosed

as an additional attribute. For example, when disclosing insurance information the insurance provider can be released so that he can be contacted to handle payment. Since not all identity providers can provide attribute values with the same degree of certainty, a Level Of Assurance [60] (LOA) can be associated with the attributes provided by identity providers and requested by service providers. Some identity providers can be authoritative with respect to certain attributes while other could be obtained during the registration process of the user. There is strong control over the exchange of information. The right to request attributes can be revoked if it is detected that a service provider abuses the requested information. Service providers are assured that the information is retrieved from trustworthy identity providers. The user can select which information is released during authentication. Further, the system can also be used in offline settings.

5.1 General Approach

This chapter proposes a privacy friendly user-centric identity management approach based on a secure element. Multiple identity providers can endorse the user's personal information to multiple service providers. The secure element is the mediator between identity providers and service providers. More precisely, an identity provider can store some of the user's personal attributes (or properties thereof) in the user's secure element. Information that is endorsed by identity providers can then be disclosed to service providers. A system-wide ontology is used that allows the unique identification of attributes and attribute properties. Service providers use the information to provide fine-grained access control and offer personalized services. For instance, when acquiring a ticket for a soccer contest, a student can request certain personal attributes from identity providers and release them to the ticketing service to get reductions. A university can vouch that the user is a registered student, while the government can attest that the user has not been convicted for hooliganism.

Our approach combines several desirable privacy features of current identity management systems. First, *an identity provider cannot profile the user's actions*, as there is no direct link between identity providers and service providers (i.e. the secure element mediates personal information requests). Moreover, collusion of identity and service providers is prevented by means of provider-specific pseudonyms. Each user has a unique pseudonym for each provider, the provider-specific pseudonym. The different pseudonyms of the same user are generated by the secure element and cannot be linked as such by any other entity in the system. Second, *the disclosure of personal information is controlled by multiple parties.* The secure element controls access

to identity information. Prior to user authentication, the service provider first has to authenticate to the secure element and prove that he is authorized to access certain personal attributes. A central authority issues rights to service providers to request a set of attributes based on the nature of the provided service. The secure element verifies the acceptability of the service provider's information request. Users can further restrict the disclosure of personal information or explicit user consent is required prior to the release of data. Each user can configure its own privacy policy. To lower the threshold for using policies, some preconfigured policies could be presented to users.

Attributes can be cached temporarily in the secure element. This makes the scheme more efficient and usable in an offline environment. A flexible and scalable revocation procedure is foreseen; even offline services can check whether or not a secure element is revoked. The mechanism is based on a validation service that regularly updates status information in the secure element. Our approach is also flexible and scalable in the sense that new service providers and identity providers can easily be added. Moreover, a service provider can also play the role of an identity provider for other service providers.

The beneficial privacy properties that this scheme provides, originate in the use of a secure element (application). Such a secure element can be for instance a smart card or a SIM card, and is issued by a trusted third party. It acts as a gateway between identity providers and service providers. Figure 5.1 provides an overview of the proposed architecture.

5.2 Design

5.2.1 Roles

Each User (U) has a Secure Element (SE), in the remainder of this chapter also referenced to as *card*. Middleware (M) is installed at the client side (i.e. host or card reader). It allows users and remote parties to interact with the card. The Service Provider (SP) offers personalized services to authorized users. The Identity Provider (IdP) returns endorsed personal attributes to the secure element which can be released to service providers. The Card Issuer (CI) issues secure elements to users. The (Re)Validation Authority (RA) can (re)validate or block secure elements. The Audit Authority (AU) grants rights to service providers to retrieve and to identity providers to provide a specific subset of identity information of the user. The Certification Authority (CA) issues certificates to the different actors.

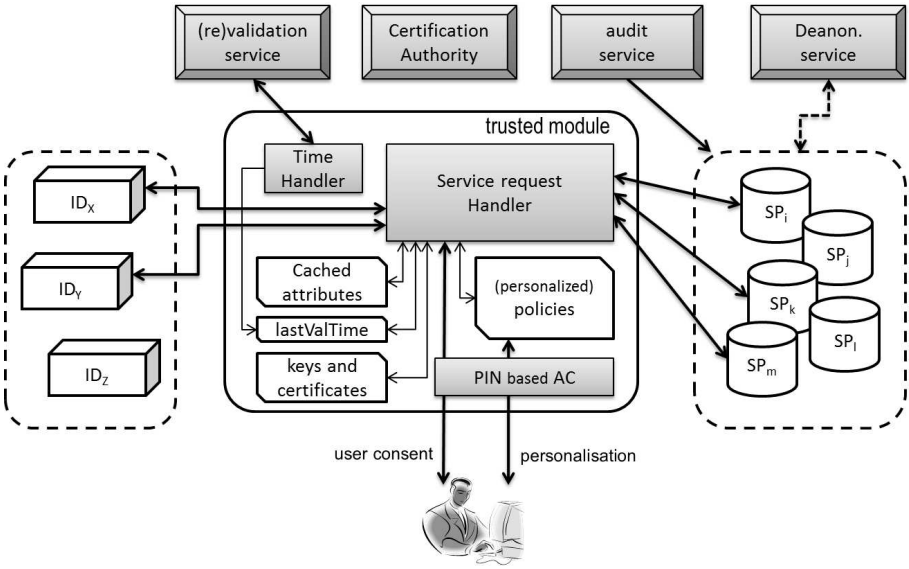

Figure 5.1: Overview of the architecture.

5.2.2 Requirements and Adversary Model

Functional Requirements

F_1 Service providers can retrieve personal attributes either stored in the card and/or managed by an identity provider.

F_2 Adding new services and identity providers is straightforward.

F_3 The card can be used online and offline.

Security and Privacy Requirements

S_1 The service provider is assured that the received attributes are reliable (i.e. meet a specified level of assurance from a trusted identity provider).

S_2 The card issuer only provides the trusted environment. This should not allow him to impersonate users.

P_1 Controlled access to personal attributes that are accessible through the card (i.e. based on rights/privileges). A trusted party, namely the audit

authority, restricts the information that can be retrieved by service providers. Users can further restrict the dissemination of personal data (i.e. which information is transferred from which identity provider(s) via the secure element to which service provider(s)).

P_2 The systems allows proving possession of a valid secure element without releasing uniquely identifying information.

P_3 The system supports the generation of mutually unlinkable pseudonyms for each provider.

P_4 The system should support conditional anonymity. This allows identifying suspects in case of abuse.

P_5 The communication between providers and the secure element should remain confidential, ensuring that attackers can't gain access to identity information by eavesdropping on the communication.

Adversary Model

With respect to the cryptographic capabilities of the attacker, we follow the Dolev-Yao attacker model [85]: attackers cannot break cryptographic primitives, but they can perform protocol-level attacks.

This scheme changes some of the trust assumptions that are inherent to traditional schemes without a secure element. The paragraph below provides a clear overview of the trust relations between all actors in the system.

- The *card issuer* is trusted by the user and providers to issue tamperproof secure elements without hardware backdoor(s) containing software (i.e. identity management applet) correctly implementing the presented protocols (see Section 5.2.5) and to correctly manage the common keys. As the card issuer has the common key pair, he could impersonate all users, even non-existent users. However, the card-specific secret from which the provider-specific pseudonyms are constructed is generated by the card during initialization and is, hence, not known by the card issuer. As such, he cannot perform directed attacks to a specific user with providers where the user is known, alleviating the required trust from the user-perspective.
- The *secure element* is trusted by the user and providers to correctly execute the specified protocols and to securely store the cryptographic data. This trust is supported by the tamperproof properties of the secure

element and the trust in the card issuer to load a correctly functioning applet on the card. Essentially, adversaries should not be able to influence the card so that it deviates from the imposed protocols. Nor should they be able to obtain direct access to the memory of the secure element to extract information. Hence, providers can trust the data that was returned by the secure element. A related discussion about trust in trusted computing platforms is given in [164, 163]. Note that the German electronic identity infrastructure [32] requires similar trust assumptions.

- The *audit authority* is trusted to formalize the trust relationships between service and identity providers regarding the exchange of user identity information. The audit authority therefore generates signed files. Each signed file allows the secure element to ensure that it only transfers information from identity providers to service providers which matches their trust relationship without needing to inform the identity provider which service provider the user is using and vice versa. Both entities need not be aware of the actual identity of the provider to or from which the information is transferred, only that both parties have a trust relationship. Section 5.2.4 discusses several approaches for realizing the audit authority.

- *Identity providers* are trusted to provide correct user information to service providers via the secure element. This trust is supported by the audit authority that assesses which attributes can be provided with which level of assurance.

- *Service providers* are trusted by the identity provider and users not to abuse the data received during authentication. This trust is supported by both the audit authority that can assess the privacy policies of the service providers and the user that controls which information is disclosed during authentication.

- The *(re)validation authority* is trusted by the users and providers to correctly execute the (re)validation protocol.

- The *workstation* is trusted by the user to provide a secure interface via which the user can enter his passcode, control the information disclosed during authentication and activate policies on the card. The user, further, trusts the workstation to implement adequate measures to anonymize the communication with providers, as discussed in Section 2.5. To decrease the trust required by the user in his workstation, a card reader with pinpad and screen can be used. Alternatively, the middleware could be implemented in a trusted execution environment on the workstation (see Chapter 7), providing a secure interface to the card. If the secure element is realized using a SIM or a Mobile Security Card [8], the middleware

can be implemented on the mobile which serves as a proxy between the secure element and the outside world. The mobile can inform the user about authenticating service providers and allow the user to control the disclosure of information. The mobile is typically more trustworthy than a random workstation on which the user might need to authenticate. Similarly to regular workstation, technologies [189, 90, 191, 206, 105, 177] exist to implement (part of) the middleware in a trusted execution environment.

5.2.3 Public Key Infrastructure

The keys and certificates that are maintained in the system are listed below:

- The public keys of *CA*s in the system are placed on the card during initialization. This allows the card to verify the certificate of the (re)validation authority, audit authority and providers.
- Each service/identity provider generates a key pair, the public key of which is certified by a certificate authority (CA) in a certificate $cert_p$.
- The audit authority has a certified key pair for assigning the rights of providers to their certificates. It generates a signed file containing the provider's privileges and a hash of the provider's certificate. This allows the secure element to enforce the correct access control policy for each provider.
- The (re)validation authority generates a key pair, the public key of which is also certified by a CA (i.e. $cert_{ra}$). This certificate grants the necessary rights to (re)validate cards.
- The cards contain a key pair (sk_{Co}, pk_{Co}) – certified by a CA – that is identical for a large set of cards. Since the private key is kept inside the trusted environment, making a valid signature with it proves that the card is genuine. Hereby, no uniquely identifying information about the card nor the card owner is released.

Note that a provider can have the rights to both provide and request user attributes.

5.2.4 Controlled Release of Information

As previously discussed, sufficient control over the flow of user information by all stakeholders is of paramount importance. Users want to be assured that the information they disclose during authentication is not abused by service providers. Service providers want to be assured that the received information is obtained from trustworthy identity providers. Further, the information that service providers can request should depend on the nature of the service they provide. If service providers can independently determine which information is requested during authentication, they may tend to request more information than necessary for the provided service. Similarly, Android applications often require more permissions than required to implement the provided functionality [112]. Users have no choice but to ignore the potential privacy implications if they want to install the application. Since many service providers like to acquire user data, there are rarely privacy-friendly alternatives. Users should also have control over the disclosure of their personal information. They should be informed about which information is released to which parties.

by the identity provider and users to assign reasonable access rights (i.e. based on the nature of the service) to service providers and to verify that service providers enforce adequate security and privacy policies to protect the user's information. Service providers trust the audit authority to assess what information an identity provider can assert, assigning different levels of assurance to different attributes. This component replaces the pairwise trust establishment process between service providers and identity providers as used in today's federated identity management model.

These control measures are partly realized via an audit authority that assigns rights to providers. These restrict the set of queries (i.e. the set of attributes or properties thereof) that can be requested from the card. Users are assured that their information is only disclosed to service providers with a valid certificate. This does not guarantee that a service provider will not abuse the information received during authentication. However, if abuse is detected, the certificate of the service provider can be revoked. For identity providers, they restrict the set of attributes that can be provided. Two different approaches to establish the audit authority are discussed below. Each option realizes different properties regarding trust, scalability, complexity and control.

- A single trusted third party (i.e. the audit authority) can be used to assign rights to service and identity providers, establishing one big federation. This authority is trusted by the identity provider and users to assign reasonable access rights (i.e. based on the nature of the service) to service providers and to verify that service providers enforce adequate security

and privacy policies to protect the user's information. Service providers trust the audit authority to assess what information an identity provider can assert, assigning different levels of assurance to different attributes.

- A second approach generalizes the previous approach and maps the trust model of federated identity management systems to the system described in this chapter. Identity and service providers can establish trust relations, founding a federation. Providers can be in multiple federations. In a federation, user information can be exchanged between providers via the secure element. Each provider in the federation can provide and/or request a specific set of attributes. Which information can be provided/requested by which providers and with which level of assurance is decided by the members of the federation. This could be coordinated by the *federation authority* which is responsible for administering the federation and could also take up the role of the audit authority. The federation authority can request an audit certificate from a *CA* and use it to assign the access rights to the members from the federation. Each federation is assigned a unique identifier that is included in the privileges file(s) of each provider in the federation and allows the secure element to enforce the restrictions bound to each federation.

Working with a single trusted third party as an audit authority is a relatively simple approach. It is transparent to users as the attributes a provider can request does not vary over federations. It is also easier to implement on the secure element as the algorithm to determine whether a user can comply with an attribute query and which identity providers need to be contacted is less complex. Moreover, as a single audit authority is used, its certificate can be cached on the secure element, minimizing data traffic. It is easy to add new providers and they immediately have access to user information without needing to negotiate and join an existing or establish a new federation. However, each provider in this federation implicitly trusts each provider in the federation to provide or request the attributes as specified by the audit authority. This might be feasible if the number of providers in the system is relatively small. However, as the number of providers in the system increases the more the trust relationships as established in the federation will conflict with the trust actual trust relationships. In these settings, the approach with multiple federations is better suited. The smaller scope of a federation gives each party a better overview of the different identity and service providers information can be respectively requested from or provided to. The decision-making process for allowing new members to the federation and assigning privileges to service and identity providers can involve all the members in the federation. This gives each member more control over the attribute aggregation process compared to working with a single audit authority.

5.2.5 Protocols

This section presents the different protocols of the system. In the protocol listings, authenticated encryption is assumed. Hence, apart from the confidentiality, also the authenticity of encrypted messages is protected.

Card Issuance

The card issuer issues cards containing the identity management applet and the required keys to users. Before an individual can use the card, it needs to be activated. During this phase, a unique card-specific secret $\mathcal{S}_C$ is generated on the card, which is later used to generate provider-specific pseudonyms. Additionally, the user has to confirm a personal PIN. During initialization, the card is also assigned a unique card-specific identifier (`chip_number`). This identifier is used for blocking revoked cards.

During card issuance, *CI* can already store a set of immutable attributes (of the card or the card holder) in the card. Alternatively, if the card is used as a *governmental identity card*, *CI* may cooperate with a governmental identity provider IdP_{Gov} to bind the card to a specific user. The card discloses the provider-specific nym_{Gov} and `chip_number` to IdP_{Gov}. This allows IdP_{Gov} to bind the citizen to the card and corresponding `chip_number`. The card can now request attributes from IdP_{Gov}. The `chip_number` enables the government to set up the (re)validation authority. If a user looses his card, the user can present himself with the (re)validation authority to block the card. Either a third-party token could be used to prove the identity of user or the user could be provided with the `chip_number` during card issuance.

(Re)Validation of the Card

The card (re)validation protocol confirms that a card is still valid at a given time. The validity of the card is verified by the (re)validation authority which keeps track of the revoked cards referred to by their `chip_number`. If a card has been revoked, *RA* will block the card so that further authentications are no longer possible. Otherwise, the `lastValTime` is updated with the current time. Whenever the card contacts a relying party, it ensures that the `lastValTime` lies within a certain time frame that is acceptable for that party (see following section). The middleware should periodically (re)validate the card to ensure that it is not necessary to execute the (re)validation protocol during authentication. This would degrade performance and allow the provider

and the (re)validation authority to collude and link user information via a timing attack.

Table 5.1 shows in detail how the `lastValTime` is updated when the card is inserted in the card reader. When the card has not been revalidated recently, user confirmation is requested[1] to start the revalidation protocol (1–6). Next, an authenticated key agreement protocol is executed. Therefore, *SE* uses $sk_{Co}/cert_{Co}$) and *RA* uses $sk_{ra}/cert_{ra}$. This step results in a shared session key K_s (7). *SE* then sends the `chip_number` (encrypted with K_s) to *RA* who, subsequently checks the revocation status of the card. (8-9). If the card is revoked, a block command is sent to the card. Otherwise, the current time is encrypted with K_s and sent to the card (10–11). Finally, upon receiving the encrypted time, the card decrypts it, and updates its `lastValTime` (12). The actual revalidation time is sent to the card, giving it a notion of time, since common secure elements such as SIM and smart cards do not have a real-time clock.

Table 5.1: The card is regularly revalidated by the revalidation authority (*SE*: Secure Element, *M*: Middleware, *U*: User, *RA*: (Re)Validation Authority).

revalidateTrustedModule():

(1)	*SE*	:	inserted in reader
(2)	$SE \leftarrow M$	:	`"Hello"`, *currentTime*
(3)	$SE \rightarrow M$	:	*reqRevalidation* := (`lastValTime` < *currentTime* - δ)
(4)	$M \rightarrow \boxed{U}$	:	**if** (*reqRevalidation*) showRevalWindow() **else** abort()
(5)	$M \leftarrow \boxed{U}$	:	**response** [*assume* `Yes`; otherwise abort()]
(6)	$M \overset{\text{BT}}{\rightarrow} RA$	:	`"RevalidationRequest"`
(7)	$SE \leftrightarrows M \overset{\text{BT}}{\leftrightarrow} RA$	:	K_s := authKeyAgreement(sk_{Co}, $cert_{Co}$; sk_{ra}, $cert_{ra}$)
(8)	$SE \rightarrow M \overset{\text{BT}}{\rightarrow} RA$	:	$\{$`chip_number`$\}_{K_s}$
(9)	RA	:	**if** (**not** isValid(`chip_number`)) sendBlockCommand(), abort()
(10)	RA	:	*time* := getCurrentTime()
(11)	$SE \leftarrow M \overset{\text{BT}}{\leftarrow} RA$	:	$\{time\}_{K_s}$
(12)	*SE*	:	`lastValTime` := *time*

Mutual Authentication Between the Card and a Provider

Mutual authentication between the card and a relying party, which can be either a service provider or an identity provider, is accomplished as follows. If the card is not blocked, the card and relying party *P* initiate a standard

[1]Note that steps 4–5 are optional. If they are omitted, the card will automatically be revalidated or blocked.

authenticated key agreement protocol (1–2). The relying party, therefore, uses its certificate $cert_p$, while the card uses the common certificate $cert_{Co}$, preserving the privacy of the user. The resulting session key is used to encrypt all messages sent between *SE* and *P*. The mutual authentication protocol is presented in more detail in Table 5.2.

The signature of $cert_p$ is verified using the public key of the CA which is stored on the card. Since some secure elements, such as smart cards, do not have an internal real-time clock, the `lastValTime` is used to verify the validity period of the certificate. To handle compromised/revoked provider certificates, four approaches can be used. First, the card can issue an Online Certificate Status Protocol (OCSP) request via the middleware. This, however, introduces a significant delay in the authentication process. Instead of using OCSP, the CRL can also be loaded on the card. Since revocation of provider certificates is typically much less common compared to revocation of end-user credentials, the secure element may have sufficient memory to contain the entire CRL. This list can, for instance, be updated during the card (re)validation protocol. A third approach is to use short lived provider certificates. Providers can periodically request a new $cert_p$ based on a long term certificate. This approach requires no revocation checks on the secure element but a vulnerability interval (i.e. time between the certificate being compromised and the expiry date) exists. Although, the reissuance of $cert_p$ can be automated based on a long lived certificate, the load on the CA infrastructure significantly increases. A fourth approach is to use the middleware to check the revocation status of the provider's certificate. This minimizes the overhead caused by the revocation check but increases the trust required in the middleware. This approach can, for instance, be used when the secure element is integrated in a smartphone.

After key agreement, a new session is started (3-4), and the session identifier *sesId*, is passed to *P* (5). The card manages session data that contains all the information related to the context of the authentication. The secure element supports multiple simultaneous sessions. This is necessary when one or more identity providers need to be contacted to resolve the attribute query of the service provider. The secure element can also maintain simultaneous sessions with service providers. *P* will now send the oldest acceptable validation time, `accValTime`, to the card (6). The card verifies if it has been revalidated more recently than `accValTime` (7). This ensures that the card was at least valid until `accValTime`, without revealing the precise value of `lastValTime`. How fast revoked cards are blocked depends on the last acceptable validation time required by the provider. The closer the last acceptable validation time is to the current time, the faster revoked cards will be blocked. The middleware should periodically (re)validate the card to prevent that the revalidation protocol should be executed during authentication since that would decrease

performance and potentially allow the (re)validation provider to collude with providers and link profiles of users. If the card has been revalidated recently enough, the card verifies the rights of the provider by checking the link with the provider's certificate and the signature by the audit authority (8). If the verification succeeds, the card stores the name of the relying party and its access rights in the temporary session object (9-10). The former can later be used to generate a service-specific pseudonym. The latter specifies each attribute (e.g. *name*), attribute group (e.g. *eHealth data*) and/or attribute property (e.g. *age* $>$ *18*) that can be requested/provided using attribute and provider identifiers.

Table 5.2: Authentication between the provider and the card (*P*: Provider, *SE*: Secure Element, *M*: Middleware).

$sesId :=$ Authenticate():			
(1)	SE	:	**if** (**not** isActivated() $\vee$ isBlocked()) abort()
(2)	$SE \leftrightarrows M \overset{\text{ch}}{\leftrightarrow} P$	:	$K_s :=$ authKeyAgreement(sk_{Co}, $cert_{Co}$; sk_p, $cert_p$)
(3)	SE	:	$sesId :=$ startNewSession()
(4)	SE	:	session[$sesId$].sKey $:= K_s$
(5)	$SE \rightarrow M \overset{\text{ch}}{\rightarrow} P$	:	$\{sesId\}_{K_s}$
(6)	$SE \leftarrow M \overset{\text{ch}}{\leftarrow} P$	:	$sesId$, $\{$ "validatedAfter", accValTime, $rights_p$ $\}_{K_s}$
(7)	SE	:	**if** (lastValTime $<$ accValTime) abort()
(8)	SE	:	**if** (**not** verifyRights ($cert_p$, $rights_p$, pk_{au})) abort()
(9)	SE	:	session[$sesId$].rights $:= rights_p$
(10)	SE	:	session[$sesId$].subject $:= cert_p$.subject

Access to (Personalized) Services

Before an individual can use a service, the service provider may require the user to release certain personal attributes either stored in the card or available from identity providers. This is realized as follows and illustrated in Table 5.3.

The card and the service provider mutually authenticate (see previous section) (1). Next, the service provider sends its `attributeQuery`-command to the card (2). This query can contain an explicit request to disclose the service-specific pseudonym nym_{SP}. Note that not every service provider should be able to link different service requests by the same user. For instance, for some sites it is sufficient to prove that you are older than 21 in order to access the service. Next, the card verifies that the service provider is allowed to query this information, based on the access rights obtained during authentication (3). User-policies may further impose restrictions on the disclosure of these attributes. Optionally, based on the user's policies the card may send the query together with ses_{SP}

and *SP*'s name to the middleware *M* to be displayed to the user. The access control policy of the service provider can also give the user the choice between different sets of attributes. For instance, for asserting that a user is older than 18, the user can choose to release his birthdate or prove possession of a valid driver's licence. By selecting the optimal policy, the number of IdPs that need to be contacted to obtain the required attributes can be minimized, speeding up the authentication process. After the user's consent, the (possibly modified) query (actually, the delta) is returned to the card (5-8). The card removes the delta from the query (9) and the PIN is verified (10). The card then checks which attributes are locally available on the card (11) and which attributes have to be fetched from identity providers. For determining which attributes should be retrieved from which identity providers to minimize the number of contacted identity providers, `attributeMap` is used. This is a data structure that defines which attributes can be retrieved from which identity providers. This strategy was proposed as a *Linking Service* in [62]. First, the attributes available locally, such as the provider-specific nym_{SP}, are gathered (12). Generating provider-specific pseudonyms can be kept very simple: nym_{SP} := $\mathsf{prf}_{\mathcal{S}_\mathcal{C}}(cert_{sp}.\text{subject})$ where `subject` is the name of the service provider as it is kept in its certificate; $\mathcal{S}_\mathcal{C}$ is a card-specific secret. Note that in case the service specific nym_{SP} is revealed during enrollment (see following section), disclosing this pseudonym may be sufficient to use the service. Second, the card authenticates to all identity providers from which data has to be fetched (using the protocol defined in the previous section), and sends the attribute request together with the identity provider-specific pseudonym nym_{IdP}. Based on nym_{IdP}, the identity provider can fetch the data from its database and return it to the *SE* (13-21). Finally, the card encrypts all requested attributes with K_s and sends them to the *SP* (22).

Offline Use. The secure element can be used for proving attributes offline. Therefore, a caching system is implemented that retains commonly used attributes or attributes specified by the user in the secure element. The identity provider can define which attributes may be cached together with a retention time. Additionally, the service may impose restrictions on the freshness of the (cached) data; such restrictions can easily be passed to the card by including them in the service provider's certificate. The caching mechanism can also significantly improve the performance of the authentication protocol. Fewer identity providers need to be contacted to resolve the attribute query if commonly requested attributes such as *name* and *address* are cached in the secure element.

Deanonymization. For certain services, it must be possible to revoke the user's anonymity in case of abuse. One strategy consists of encrypting the `chip_number` together with the service-specific pseudonym or a random number

Table 5.3: The card releases attributes to the authenticated service provider (*SE*: Secure Element, *M*: Middleware, *SP*: Service Provider, *IdP*: Identity Provider).

Identify():

(1)	$SE \leftrightarrows M \overset{\text{ch}}{\leftrightarrow} SP$:	ses_{SP} := Authenticate()
(2)	$SE \leftarrow M \overset{\text{ch}}{\leftarrow} SP$:	ses_{SP}, $\{$ `"attributeQuery"`, $query\}_{\underline{\texttt{session}}[ses_{SP}].sKey}$
(3)	SE :	**if** (**not** verifyQuery($query$, $\underline{\texttt{session}}[ses_{SP}]$.rights)) abort()
(4)	SE :	$query^*_P$:= applyPolicy($query$)
(5)	$SE \rightarrow M$:	$query^*_P$, ses_{SP}, $SP = \underline{\texttt{session}}[ses_{SP}]$.subject
(6)	$M \rightarrow$ [U] :	showQueryWin(SP, $query^*_P$)
(7)	$M \leftarrow$ [U] :	**response** [Assume **OK** [$delta^*_U$, **PIN**]; otherwise abort()]
(8)	$SE \leftarrow M$:	`"deltaQuery"`, ses_{SP}, $delta^*_U$, **PIN**
(9)	SE :	$query^*_{P,U}$:= $query^*_P - delta^*_U$
(10)	SE :	**if** (PINincorrect(**PIN**)) handleWrongPIN()
(11)	SE :	$attsPerIdP$:= resolveQuery($query^*_{P,U}$, `attributeMap`)
(12)	SE :	$atts$:= getLocalData($attsPerIdP$ [`"local"`])
(13)	SE :	**forall** (IdP **in** $attsPerIdP$):
(14)	$SE \leftrightarrows M \overset{\text{ch}}{\leftrightarrow} IdP$:	ses_{IdP} := Authenticate()
(15)	SE :	nym_{IdP} := getNym($\mathcal{S}_\mathcal{C}$, $\underline{\texttt{session}}[ses_{IdP}]$.subject)
(16)	SE :	qry := makeQuery($attsPerIdP$ [IdP])
(17)	$SE \rightarrow M \overset{\text{ch}}{\rightarrow} IdP$:	$\{nym_{IdP}, qry\}_{\underline{\texttt{session}}[ses_{IdP}].sKey}$
(18)	IdP :	$atts_{IdP}$:= getData(qry)
(19)	$SE \leftarrow M \overset{\text{ch}}{\leftarrow} IdP$:	$\{atts_{IdP}\}_{\underline{\texttt{session}}[ses_{IdP}].sKey}$
(20)	SE :	$atts$.add($atts_{IdP}$)
(21)	SE :	**endfor**
(22)	$SE \rightarrow M \overset{\text{ch}}{\rightarrow} SP$:	$\{atts\}_{\underline{\texttt{session}}[ses_{SP}].sKey}$

(in case no pseudonym is revealed) using the public key of a trusted third party entitled to deanonymize the user under certain circumstances. The `attributeQuery`-command can include a request to disclose the encrypted `chip_number`. In case of abuse (and if the deanonymization option was used), the service provider forwards the encrypted data together with proof of the abuse to the trusted party, which can decrypt the data and use the `chip_number` to obtain the user's real identity from the card issuer. This approach results in minimal overhead, but only links the identity of the user to the user authentication itself. Hence, any actions by the user after authentication cannot be provably linked to the deanonymization data. This complicates the deanonymization process. To realize transactional accountability, a unique private key and certificate can be included in the card. The card can sign a hash of the transaction data with that private key. The signature and certificate are encrypted with the public key of the deanonymization authority, which results

in the deanonymization data. This links the identity of the user to a transaction and allows the deanonymization authority to verify that the transaction violates the service provider's terms. For instance, an anonymous user posting racist messages on an forum or making available copyright protected files via a public storage provider. Moreover, since every card holds a unique private key, the deanonymization data is provably linked to one user. This system, however, increases the trust required in the middleware. The user needs to trust the middleware that the transaction shown by the middleware is the one signed by the card.

An alternative strategy consists of extending the responsibilities of the (re)validation authority with deanonymization. The card can contact the (re)validation authority to retrieve a deanonymization attribute. The card releases `chip_number`, after which the (re)validation authority gets the user's identity from his database and signs an encryption (with the public key of the (re)validation provider) of the user's identity together with a timestamp. The signature, encrypted identity and timestamp are then released to the service provider, via the card. The service provider can check the validity and freshness of the deanonymization data via the timestamp and signature. The (re)validation authority can decrypt the encrypted identity of the user using his private key. The advantage compared to the previous approach is that it is harder for attackers who hacked a card to obtain deanonymization data as the (re)validation authority will not issue deanonymization attributes for revoked cards. Attackers who hacked their own card can only obtain deanonymization data containing their own identity information as a valid `chip_number` is required. The `chip_number` is sufficiently long to mitigate the impact of brute force attacks.

Enrollment

Similar to accessing personalized services, a user may enroll with an identity provider. During enrollment the card discloses the identity provider-specific nym_{IdP} and possibly attributes from other identity providers. During enrollment, the identity provider links nym_{IdP} to its profile of the user. The same pseudonym is used during future requests for personal information by *SE*. When a connection with the secure element is established, the identity provider can update `attributeMap` based on the attribute information it can provide about the user.

Personalization

Users can define and submit personal policies to the card. The user's PIN is used to authenticate the policies to the card. Subsequently, the card automatically enforces the active policies. Service and identity providers can be blacklisted or whitelisted, or a combination thereof. If whitelisting is used, new service and identity providers have to be added explicitly to a whitelist in the card. Users can further select a set of attributes that should always be cached on the card.

5.3 Evaluation

This section first matches the solution with the requirements defined before, followed by a discussion and a comparison with related work.

5.3.1 Requirements Review

Functional Requirements

Functional requirements F_1 and F_3 are realized by a caching table on the card. The table keeps a set of personal attributes and their retention time. The identity provider defines the validity interval. The retention time of the owner's `name` may be unlimited whereas the retention time of `student related information` may be limited (i.e. one year). Caching attributes allows offline use of the card. It, however, does require that they have been fetched from identity providers before offline use. Note that mutual authentication and generating provider-specific pseudonyms does not require communication with external entities. Adding new identity and service providers to the system is straightforward. After receiving a certificate from a *CA*, user can enroll and authenticate (cf. F_2). Upon joining one ore more federations, depending on the model used to realize the audit authority, the provider obtains the required privileges to request and or provide a specific set of attributes.

Security and Privacy Requirements

Only genuine secure elements contain the common key pair to establish the secure channel with providers using standard authentication, encryption and integrity protection protocols. The tamperproof properties of the secure

element ensure that this confidential information cannot be easily extracted by an attacker. Hence, service providers are assured that the information obtained over the secure channel originates from a genuine secure element. As the card issuer is trusted to load an applet on the secure element that correctly implements the desired functionality and identity providers are certified by an audit authority, the service provider is assured that only information obtained from trusted identity providers (i.e. in the same federation(s) as the service provider) with adequate level of assurance is released. A PIN is required to activate the secure element and the card (re)validation protocol allows blocking lost or stolen cards, preventing abuse. This satisfies security requirement S_1. During card issuance and activation, the card issuer obtains only a minimal set of information (cf. S_2). It cannot calculate the card's pseudonyms, which prevents impersonation of a specific person by the card issuer.

Multiple measures have been taken to realize the privacy requirements. In contrast to many identity management systems used in the public domain, this solution does not require direct communication between identity providers and service providers. On the contrary, identity providers are unaware about the services that are consumed by the card holders. To ensure that timing attacks cannot be used to link the (re)validation process to the authentication, the middleware should periodically initiate the (re)validation protocol.

An audit authority assigns access rights to service providers. These access rights can be coarse-grained, such as *access to all eHealth data of an individual*, or fine-grained, such as *only allow the verification of certain age properties (e.g. age $>$ 18)*. Moreover, if several trusted identity providers can supply the requested information, the service provider does not know which identity provider was used. For instance, when a service provider queries whether the card owner is a student, proving that property does not reveal the university providing this information. Hence, the audit authority ensures that service providers can only issue queries related to the nature of the provided service. The access privileges of maliciously behaving providers can be revoked. Additionally, card personalization and user-consent give the user control over the disclosure of personal attributes. For instance, the user could specify which identity providers are used to retrieve the required information. The combination of these features realizes strong distributed control over the release of identity information, satisfying requirement P_1. The system uses a common key pair to prove possession of a genuine secure element, releasing no uniquely identifying information (cf. P_2). The card can generate unique, mutually unlinkable pseudonyms for each provider based on a unique card-specific secret (cf. P_3). Users remain anonymous if they do not reveal the service-specific pseudonym nor any other uniquely identifying information during authentication or via context information (as discussed in Section 2.5). Deanonymization allows

enforcing accountability measures as required by P_4. The communication of identity information between the secure element and providers occurs over a secure channel, protecting the confidentiality of the data using standardized protocols. This satisfies privacy requirement P_5.

5.3.2 Discussion

The proposed scheme has several advantages but also constraints. In the following, we discuss the most important ones and show how to tackle them.

Security

The main disadvantage with respect to the security of the system compared to anonymous credentials and other identity management systems is the lack of binding between the private key of identity providers and the attribute values released to service providers. Malicious users with sufficient resources could try to break the tamper resistance and extract the common private key from the secure element. This introduces potential threats to the users and providers in the system.

- *Users:* Even if the common private key of one user is compromised, an attacker cannot access profiles of users with service providers. An attacker needs to know the service-specific pseudonym of the user. Since this is generated using a secret value generated by and stored on the secure element of the user and transmitted encrypted to the provider, an attacker cannot impersonate users other than the owner of the compromised card. Brute force attempts are prevented with sufficiently long pseudonyms. An attacker could, however, release forged identity information when requesting access to a service.
- *Providers:* Service providers rely on the tamper resistance of the card to prevent users from directly using their private key to release fake information. However, if a common key is compromised an attacker can forge the information released to a service provider, hereby compromising the integrity of the service. If abuse is detected, the common key that is used needs to be revoked. This affects each user with a secure element containing the same common key. Each of these secure elements needs to be replaced after the revocation of the common key.

Two complementary approaches to mitigate the consequences of hacked secure elements are discussed below.

- The first approach relies on identity providers timestamping and signing the attribute values released to the card. Since possession of the common private key is not sufficient for malicious users to eavesdrop the communication channel between the card and identity providers of other users it tackles the problem of malicious users using fake information or information from other users to gain access to personalized services. It, however, also introduces several disadvantages to the system. For instance, service providers learn with which identity providers users are enrolled. This makes it easier for identity and service provider to collaborate and link user profiles based on timing information. The strategy also has an impact on other architectural decisions. For instance, the caching algorithm needs to be modified since either all or no attributes obtained from an identity provider need to be cached on the card due to the signature. Further, if cached attributes are released, all attribute values asserted by the signature need to be released while only a subset might be required. Identity providers could sign individual attribute values. This strategy, however, increases the communication and storage overhead with a digital signature for each fetched or released attribute.

- To decrease the potential benefits of extracting a private key from a smart card, a mechanism can be built in to discourage users to share or sell the secret.

  To incorporate this approach in the current prototype, two major modifications need to be made. First, the card is now equipped with an anonymous credential, instead of a public-key credential, containing a validity period. Hence, after initiating a secure session with the provider based on the provider's certificate, the card uses the anonymous credential to prove that it is a genuine card and releases the validity period of the credential. Short lived credentials are used to avoid the usage of complex credential revocation techniques. The (re)validation protocol now performs a credential update during which the validity period is updated. The service-specific pseudonym is now generated based on the master secret of the anonymous credential. Hence, when malicious users extract their private key from the smart card they are discouraged from sharing or selling it since this allows access to all personal services of the user (e.g. online tax declaration) as opposed to the current approach where service-specific pseudonyms are generated based on a separate master secret that does not need to be shared. Compared to an identity card based on full fledged anonymous credentials, this approach is more flexible and allows more efficient proofs, as only a basic credential with a limited number of attributes is used.

The lack of attribute certification remains the most important attack vector. However, the strategy described above limits the impact and hence, decreases the incentive to hack the card. Using an anonymous credential will deteriorate the performance. A hybrid system in which the anonymous credential is only used for authentication towards service providers can mitigate the performance impact.

Revocation

In the proposed system, card (re)validation and blocking are handled by a dedicated party. This is a significant advantage for service (and identity) providers since they no longer need to maintain certificate revocation lists or support OCSP responders from different CAs, especially since mobile tokens are easily lost or stolen. For instance, according to a Belgian newspaper article [176], approximately 200 000 electronic identity cards were lost or stolen in 2009. All non-activated cards are also contained in the revocation list. This results in large revocation lists. Moreover, security threats resulting from outdated certificate revocation lists are avoided. However, sensitive services should require a recent `accValTime` resulting in a short vulnerability window when credentials are compromised. Less sensitive or offline services could accept a more permissive window of vulnerability.

Provider-specific pseudonyms are generated based on a unique secret generated on the secure element. However, if the secure element is revoked because it is lost or stolen, a user should be able to generate the same provider-specific pseudonyms using a new secure element. Therefore, the system should provide a mechanism to backup the pseudonym. The card issuance phase can be extended to allow the user or card issuer to obtain and restore the (password-encrypted) pseudonym. Note that if the card issuer holds the unencrypted pseudonym, he can generate all the provider-specific pseudonyms of the user.

5.3.3 Comparison with Existing Systems

The proposed approach is compared to three existing systems namely an Idemix-based user-centric IdM system [34], Shibboleth and the German electronic identity card (GeID), each representing a different category of identity management systems as described in Chapter 2. The results of the comparison are summarized in Table 5.4.

Both Shibboleth and the German eID only allow the user to use one identity provider during authentication with service providers. Shibboleth, however,

allows the user to be registered with multiple identity providers, potentially increasing the diversity in user information that can be requested. In both the Idemix-based system and the approach presented in this chapter, the user can release information obtained from multiple identity providers during authentication.

In offline settings, no third parties can be contacted during authentication. Since Shibboleth delegates authentication to a third party (i.e. the identity provider) it does not meet the requirement for offline systems. The three other systems can be used in offline settings.

Two aspects of controlled attribute release are distinguished. The first aspect is the *specification* of access control rules to the identity information of the user by one or more entities in the system. The second aspect is the *enforcement* of these access control rules by an entity in the system. One or more parties can specify which information may be provided to which service providers. The party who specifies the access control rules is, however, not necessarily also responsible for enforcing these rules. In the German eID system, only certified service providers can request information from the card. The user can also restrict the information that can be requested. These rules are enforced by the identity card. In the Shibboleth system, the identity provider and user are responsible for specifying, the identity provider for enforcing the access control policy. In the Idemix-based system, the middleware is responsible for enforcing the user-specified access control rules. In our approach the audit authority assigns rights to identity and service providers. The rights of malicious service providers to request attributes can be revoked. Service providers are assured that attributes are only retrieved from trusted identity providers. Further, the user can specify restrictions on which information can be disclosed to which service providers. These rules are enforced by the secure element of the user.

In the German eID, the Idemix-based system and our approach unlinkable service-specific pseudonyms can be generated during authentication with service providers. This limits the ability of service providers to combine their databases and obtain extensive user profiles. In the Shibboleth system, however, the identity provider learns which service providers a user accesses and knows the pseudonyms of the user. Shibboleth, however, does allow the user to release unique pseudonyms towards service providers that are mutually unlinkable.

In the Shibboleth and the Idemix-based system, attributes are digitally signed by the identity provider indicating trustworthiness of the released information. In the German eID and in the architecture presented in this chapter, the released attributes are not digitally signed. Instead of using digital signatures, these systems rely on the tamperproofness of the hardware and the correctness of the hardware and software to ensure the trustworthiness of the released

information.

In the German eID service specific revocation lists are generated and distributed to the service providers. This introduces a significant overhead in settings with many service providers. In the Shibboleth system, the revocation status is verified by the identity provider and depends on the technology used during authentication between the user and the service provider (e.g. username/-password, X.509 credential). Although many revocation schemes exist for Idemix credentials, currently no scheme enables efficient revocation in large scale systems [128]. In our approach, an efficient card validation protocol is implemented to disable revoked cards. However, if the card is compromised, the card's credential needs to be revoked using standard revocation lists. Since common keys are used for a batch of cards, the entire batch needs to be replaced with new cards.

Table 5.4: Comparison between the system proposed in this chapter and existing identity management systems.

	GeID	**Shib.**	**Idemix**	**Our approach**
Attr. aggregation	No	No	Yes	Yes
Offline settings	Yes	No	Yes	Yes
Attribute release				
Specification	AU+U	IdP+U	U	AU+U
Enforcement	TM	IdP	MW	TM
Profiling	Limited	Yes (IdP)	Limited	Limited
Attr. assurance	Trusted HW	Sig.	Sig.	Trusted HW

5.4 Conclusion

This chapter presented an approach for a user-centric identity management system using secure elements. We demonstrated the high flexibility and good privacy and security properties of the proposed architecture. The system does not rely on anonymous credential systems to realize privacy-preserving features such as selective disclosure and pseudonymous authentication. Instead, the public-key credentials and the tamperproof property of the secure element are used to assert the validity of user information. This makes the performance of our system less dependent on the number of attributes supported by the system than systems relying on anonymous credentials for these features. The release of personal information is controlled at two levels. An audit authority defines system-wide access rules. Users can further restrict those access

rules. The main issue when using trusted hardware components is secure user interaction. Since most secure elements do not allow direct interaction with the user, a terminal device is required. Recent technologies focus on establishing a trusted execution environment on commodity devices, such as smartphones [189, 90, 191, 206, 105, 177] and workstations (see Chapter 7). These can be used to realize a trustworthy interface with the secure element or even an entire virtualized secure element [48], potentially considerably improving the performance.

Chapter 6

User-Centric Identity Management: Validation and Case Studies

The previous chapter proposed a privacy-friendly user-centric identity management approach based on a trusted secure element. The secure element can aggregate attributes from multiple identity providers and, subsequently, provide them to service providers. Service providers can enforce fine grained access control policies based on the user attributes obtained during authentication. The system supports several desirable features such as attribute aggregation, selective disclosure and multishow unlinkability. The information released to service providers is controlled by several parties. First, an audit authority mandates which attributes can be provided by an identity provider and requested by a service provider. Second, the user authorizes the authentication via a PIN.

Where the previous chapter described the architecture of the identity management system and presented a theoretical evaluation, this chapter illustrates the feasibility of the presented system by presenting a proof-of-concept implementation. The proof-of-concept implementation consists of several reusable software components that facilitate the integration of the IdM system in practical case studies. Two case studies are presented in the second part of this chapter, validating the identity management system.

6.1 Proof-of-Concept

This section presents a proof-of-concept implementation of the identity management architecture presented in Chapter 5. The proof-of-concept uses a single audit authority that assigns rights to service and identity providers to respectively request and provide attributes. The provider's privileges are embedded in $cert_p$. A single CA is used that generates the providers' certificates based on a report from the audit authority stating which attributes can be requested or provided. The public key of the CA is stored on the secure element during initialization. The proof-of-concept relies on the middleware to verify the revocation status of the providers' certificates.

The proof-of-concept uses a Mobile Security Card [8] from Giesecke & Devriendt to realize the secure element functionality. The Mobile Security Card contains a tamperproof smart card chip but has the form-factor and interface of a µSD. This allows it to easily interface with a wide range of devices (e.g. laptops, mobile phones and tablets), contrary to contact smart cards that require a dedicated reader. This increases the flexibility for deployment. Many smartphones provide a µSD slot. It can be plugged in to workstations and laptops via a µSD to USB adapter. This is the setup used for the proof-of-concept. Giesecke & Devriendt provides a Windows driver that registers the Mobile Security Card as a smart card with the operating system. Applications running on the operating system can, hence, interface with the Mobile Security Card like a regular smart card.

6.1.1 Protocols

This section discusses how the identity management protocols as presented in the previous chapter are realized in the proof-of-concept.

Authenticated Key Establishment

The authentication and (re)validation protocols rely on an authenticated key agreement protocol to establish a secure authenticated session between the secure element and a remote party (i.e. service provider, identity provider or (re)validation provider). The proof-of-concept uses the *Full Unified Model* elliptic curve Diffie-Hellman scheme from the NIST Special Publication 800-56A [156] to establish a secure authenticated session between the secure element and a provider. Using elliptic curve cryptography ensures optimal performance on the resource constrained secure element. The protocol relies on a static

certified elliptic curve key pair for both parties (i.e. $sk_{Co}/cert_{Co}$ and $sk_p/cert_p$) and an ephemeral key pair generated on-the-fly (i.e. $sk_{eph,se}/pk_{eph,se}$ and $sk_{eph,p}/pk_{eph,p}$). Both parties transfer their static (i.e. $cert_x$) and ephemeral public key to the other party. Subsequently, both parties generate two shared secrets, one certified and one ephemeral, using the Elliptic Curve Cryptography Cofactor Diffie-Hellman (ECC CDH) [129] primitive. The session key is then generated by applying a cryptographic hash function on the concatenation of the two shared secrets with a string containing public information of both parties (obtained from the certificates) and a counter to ensure that a session key of sufficient length can be generated. In the proof-of-concept, the RIPEMD-160 [82] hash function is used. The protocol realizes several desirable security properties such as key control and forward secrecy.

The proof-of-concept uses 192-bit elliptic curve keys. The group parameters *prime192v1* as specified in ANS X9.62 [21] are used. After an authenticated session key is established between the secure element and a provider, all messages are encrypted using 128-bit AES. A message authentication code is generated for each ciphertext using the HMAC [126] algorithm with RIPEMD-160 [82]. This realizes authenticated encryption [31], protecting the confidentiality, integrity and authenticity of the messages. The CA uses a 2048-bit RSA key pair to sign certificates. The CA does not use an elliptic curve key pair since RSA signature verification is faster than ECDSA verification for similar security levels [71].

Authentication

Contrary to the revalidation protocol, the authentication protocol is slightly modified compared to the description in the previous chapter. It is modified to support a challenge-response protocol flow, as shown in Figure 6.1. The authentication protocol as described in the previous chapter requires several exchanges between the secure element and the service provider to authenticate the user. However, for many case studies it is more convenient to support a single challenge-response flow. The provider sends a challenge to the secure element, containing all the information for the secure element to generate the session key (i.e. $cert_p$ and $pk_{eph,p}$), verify the authenticity and privileges (i.e. $cert_p$) and gather and provide the required attributes (i.e. *query*). The response contains the required information for the provider to generate the session key (i.e. $cert_{Co}$ and $pk_{eph,se}$), decrypt the attribute response from the secure element (i.e. $\{\texttt{accValTime}||atts\}_{K_s}$) and verify the authenticity (i.e. $cert_{Co}$ and `accValTime`). As the `accValTime` is no longer sent from the provider to the secure element over the secure channel, the secure element

now sends the used `accValTime` to the provider over the secure channel. The provider is, hence, assured that the correct time was used by the secure element.

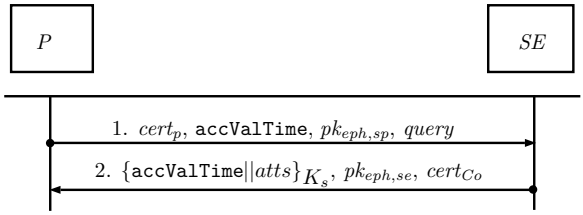

Figure 6.1: User authentication protocol (*P*: Provider, *SE*: Secure Element).

6.1.2 Software

The proof-of-concept consists of five applications. The functionality of each of the four actors in the system (i.e. secure element, identity provider, service provider and (re)validation provider) is each realized in a separate application. The fifth application is the middleware that interacts with the secure element, providers and the user to facilitate the execution of the IdM protocols. It also provides the user with an interface to manage his secure element and control the authentication. It also checks the revocation status of the provider's certificate.

The secure element application was implemented using the Java Cards 2.2.2 [5] framework. This framework allows developing applications for several types of smart cards and derived technologies such as the Mobile Security Card [8] used in the proof-of-concept. It supports the required cryptographic primitives to implement the IdM protocols. Since the Java Card framework supports only a subset of the Java language, several features available in Java are not available in the Java Cards language. Hence, most Java libraries cannot be used for the implementation of the secure element application. The other applications are realized in Java. Each application is discussed in more detail in the remainder of this section, but first several reusable data structures and software components are presented. These components facilitate the deployment and integration of the identity management architecture in applications. The Java Cryptography Extension (JCE) and Bouncy Castle [59] frameworks are used for to realize the required cryptographic operations.

Java Software Modules and Data Structures

Figure 6.2 gives an overview of the data structures and software modules that can be used by application developers to implement middleware and service/identity/(re)validation provider applications. The data structures are shown in **bold**, the software modules are represented by rectangles containing *italic* text. This section discusses each data structure and software module in more detail.

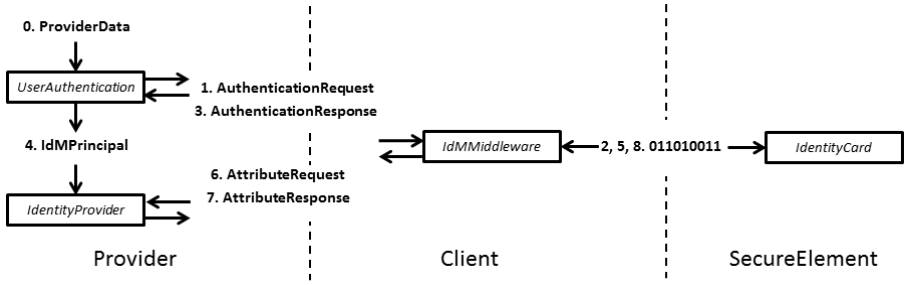

Figure 6.2: Java software modules and data structures.

CVCertificate implements a custom certificate format to avoid parsing X.509 certificates on the secure element. X.509 is a complex data structure, implementing the logic to parse it on the secure element would increase complexity and lower performance. Moreover, several custom fields need to be added to contain the information regarding the attributes that can be requested or provided. A simple custom Card Verifiable Certificate (CVC) format was defined that contains fields for all the required information and is easily parsed. Apart from the standard information contained in a certificate (i.e. issuer, subject, serial, validity interval and signature), the CVC also contains two lists of attribute identifiers, one list for the attributes that can be provided and one for the attributes that can be requested. The secure element uses standard X.509 certificates to authenticate towards providers since the common certificates do not need to contain custom information. Moreover, the standard is supported by most cryptographic frameworks.

ProviderData wraps all the data required to initialize a provider (i.e. `CVCertificate` cert, `ECPrivateKey` privKey and `Long` accValOffset).

AuthenticationRequest wraps the data transferred from the provider to the secure element during the first phase of the authentication protocol. It contains the `CVCertificate` of the provider, an ephemeral ECC public key (`ECPublicKey` ephKey), the query containing the list of attribute identifier (`ArrayList<Short>` query) and the earliest acceptable (re)validation time (`Long` revTime).

AuthenticationResponse wraps all the data transferred from the secure element to the provider during the second phase of the authentication protocol. It contains the ephemeral public key of the secure element (i.e. `ECPublicKey` ephKey), the common `X509Certificate` of the secure element and the encryption (i.e. `byte[]` encMessage) of the concatenation of the earliest acceptable (re)validation time with the user's attributes.

Attribute wraps the data related to a user attribute. It contains the attribute identifier (`Short` identifier), the attribute value (`byte[]` value) and the retention offset (`Short` retOffset). The latter indicates the maximum number of a days an attribute should remain cached on the secure element. A retention offset of zero means that it should not be cached on the card. This is only relevant for identity providers. The `Attribute` class also contains a static bidirectional map that links attribute identifiers with a human readable `String` representation.

IdMPrincipal wraps the attributes of a user (`HashMap<Short,byte[]>` attributes) and the session key (`SecretKey` sessionKey) agreed with the secure element during the authenticated key establishment.

UserAuthentication implements the logic for providers to authenticate users. It is initialized with `ProviderData`. After initialization the provider can use it to generate an `AuthenticationRequest` and verify the corresponding `AuthenticationResponse`. Upon successful verification, the module returns an `IdMPrincipal` object from which all the requested attributes and the session key agreed with the secure element are available.

AttributeRequest wraps the encrypted attribute identifiers (`byte[]` encAttributeIdentifiers) of the attributes requested by the secure element to the identity provider.

AttributeResponse wraps the encrypted attributes and their maximum retention time on the secure element (`byte[]` encAttributes) provided by the identity provider to the secure element in response to an `AttributeRequest`.

IdentityProvider constructs an `AttributeResponse` object based on an `AttributeRequest` and a `IdMPrincipal` object. The session key and attribute information contained in the `IdMPrincipal` object are used to decrypt the attribute request and generate the encrypted response. The identity provider application is responsible for retrieving the attributes of the user from storage, determining the maximum retention time and populating the `IdMPrincipal` object before passing it to `IdentityProvider`.

IdMMiddleware interprets the `AuthenticationRequest` object and generates the necessary APDU[1] messages to send to the secure element and construct an `AuthenticationResponse`. Since object serialization is not supported by the Java Card framework, explicit methods need to be implemented to serialize and deserialize objects. A Tag Length Value (TLV) data structure is used to represent the data. The data of each entry is preceded by a tag and its length. For more complex data structures (e.g. `CVCertificate`), a nested approach is used. This allows the secure element and client application to reconstruct the data object(s) from the exchanged APDUs. The primitive data types such as `Short` and `Long` are converted to their big endian representation. The `Strings` are converted to bytes using UTF-8 encoding. The point of the curve comprising the public key is represented as an octet string in compressed form as defined in ANSI X9.62 [21]. This allows serialization of all the data structures sent to and received from the secure element. If identity providers need to be contacted to resolve an attribute query, the secure element sets a specific status flag in an APDU. This triggers the middleware to query the secure element for the address of the identity provider and, subsequently, notify the middleware application that an identity provider needs to be contacted.

Secure Element Application

This section discusses several implementation aspects of the secure element application. Since secure element does not have an embedded real-time clock, the `lastValTime` is used to verify time constraints. The secure element application authenticates the user via a four digit PIN.

[1] An APDU is the communication format between a smart card and the off-card applications.

Personalized Policies. The user can manage his secure element using his PIN. First, the user can select a set of attributes that should remain cached on the card. When the retention time is expired, the secure element notifies the middleware to retrieve them again from an identity provider. Second, the user can assign a trust level to service providers: *untrusted*, *default* and *trusted*. Requests from untrusted service providers are blocked. In the *default* policy, user confirmation is required before the attributes are released. If a service provider is *trusted*, the user is no longer involved in the attribute disclosure. The query, however, is still verified using the privileges listed in the CVC.

Attribute Aggregation. The secure element application manages a set of `Attribute` objects. Each object wraps an attribute identifier, attribute value, retention time, persistence flag and the time it was last used. It can be extended with additional values such as a LOA and the identity provider from which it was obtained. The latter can be used to add support for multiple federations. Some attribute values are cached on the card, the identity provider indicates if an attribute can be cached on the card and what the retention time is. Other attribute values need to be fetched from identity providers. Therefore, each card keeps a list of `IdentityProvider` objects. These objects contain the contact information of the identity providers with which the user is enrolled and which attributes each identity provider can provide. During enrollment, the identity provider initializes the list of attributes which it can provide. The card verifies whether or not this violates the rights contained in the identity provider's certificate. This list can also be updated afterwards.

When an attribute query is received, the query handler first searches the cache. The remaining attributes are fetched from identity providers. The handler selects a minimal set of identity providers to supply the remaining attributes, ensuring optimal performance.

Memory Management. Smart cards and derived technologies typically have limited memory. The Mobile Security Card used for the prototype has around 70K bytes of available EEPROM. Moreover, the Java Card virtual machine does not implement a garbage collector, nor is it possible to explicitly release memory. Therefore, all required memory should be allocated during deployment of the program and continuously reused to avoid running out of memory during the lifetime of the card.

Caching Attributes. A fixed set of byte arrays of variable length (to minimize memory fragmentation) is allocated to store attribute values. These arrays are embedded in the `Attribute` objects that also keep the context information such as the attribute identifier, retention time and time of last usage. For an

optimal implementation, the distribution of the average length of each type of attribute should be calculated. When an attribute value is fetched from an identity provider, it might be necessary to remove another attribute from the cache. The following selection strategy is applied. First, the `Attribute` objects with the smallest memory footprint still large enough to keep the new attribute value are selected. Subsequently, the least recently used attribute value is replaced. Persistent attributes are not considered by the cache update policy. This caching strategy is straightforward while limiting fragmentation.

Static Memory Configuration. Since all memory allocations occur when the applet is deployed on the card, the maximum attribute query length, the maximum number of supported identity providers, cached attributes etc. are fixed. The initialization attributes can be used to define the amount of memory assigned to the different parts of the program when installing the applet. For instance, one can opt for allocating only a limited amount of memory for identity providers while increasing the attribute cache. During initialization, several system parameters such as the group parameters and the public key of the CA is also sent to the secure element together with the unique `chip_number` and a PIN.

Anonymous Subscriptions. For some applications such as news sites it is not necessary for a user to disclose a persistent identifier. News sites only need to verify whether the user has a subscription that allows him to view the requested content. To support this, a `Subscription` object is initialized during enrollment of the user with the service provider. The `Subscription` object contains the *id* of the service provider, a *validity period* and a *type*. The *type* allows the service provider to verify whether the requested service (or content) is included in the subscription of the user. Note that the actual *validity period* does not have to be released but can be verified by the card using the current time provided by the service provider. The card can then return either *valid* or *invalid*. The *type* field constraint can also be verified by the card but is less critical if released since typically only a limited number of subscription types are available. A pseudonym that allows the user to retain his subscription when re-enrolling with a new secure element (e.g. when the previous card was lost or expired) could be released during enrollment.

Service Provider Application

A proof-of-concept service provider is written that uses the `UserAuthentication` class to authenticate the user. The class is initialized with the required parameters and used to generate the authentication challenge and verify

the authentication response. The `ProviderData` required to initialize the `UserAuthentication` is read from an XML file. The service provider starts a server sockets that accepts incoming connections and requires the user to provide several personal attributes before granting access. An `AuthenticationRequest` is sent to the client, together with a list of attribute identifiers that are optional to release. This can, for instance, be useful in a user registration phase. The client responds with an `AuthenticationResponse`. Java object serialization is used to transfer the objects between the client and the server. The resulting `IdMPrincipal` provides the necessary information to check if the user meets the access control policy.

Identity Provider Application

The identity provider application extends the service provider application with attribute provisioning. During authentication, the user is required to release the provider specific pseudonym. This allows the identity provider to fetch the user's attributes from local storage. A simple XML file is used to store a set of user attributes for each user. Apart from the `AuthenticationResponse`, the secure element also sends an `AttributeRequest`, containing an encrypted list of attribute identifiers, to the identity provider via the middleware. The identity provider uses the session key contained in the `IdMPrincipal` object obtained during user authentication to decrypt the attribute request. The requested attributes are obtained from local storage, encrypted with the session key and sent back to the client in the `AttributeResponse`. Two instances of the identity provider application are deployed. The educational identity provider can provide the *name*, *student status* and *study domain*. The governmental identity provider can provide the *name*, *postal code*, *street and number*, *municipality*, *gender* and *place of birth*.

Revalidation Provider Application

The (re)validation provider application extends the identity provider application. Instead of providing attributes, the `AttributeResponse` contains a new `lastValTime` encrypted with the session key. The (re)validation provider application holds a certificate containing an entry that gives the provider the rights to update the `lastValTime` on the secure element and request a special attribute, the `chip_number`. The latter allows verification of the revocation status before updating the `lastValTime`.

Middleware Application

The middleware interfaces with the different actors in the system. It allows the user to connect to service providers and obtain a personalized service. To authenticate the user towards the service provider, the middleware connects with the secure element through the PC/SC API available in Java as of version 1.6. The middleware connects with service, identity and (re)validation providers over sockets. The middleware uses the `IdMMiddleware` component to interpret and construct the required messages. It, further, informs the user about the pending authentication and allows the user to enter his PIN. The user can, further, select which optional attributes to disclose during authentication towards a service provider. The selected optional attributes are added to the query in the `AuthenticationRequest` object. A certificate revocation list is used to check the revocation status of the providers' certificates.

6.1.3 Evaluation

The generic software modules and data structures can be used to facilitate the deployment and use of the IdM system in applications. The software modules abstract away the details of the IdM protocols. The applications are only responsible for exchanging the constructed messages and initializing the software modules with the required data. The proof-of-concept relies on the middleware to check the revocation status of the provider's certificate. This requires the user to trust the middleware. Other approaches, as presented in the previous chapter, lower the trust required in the middleware but introduce additional overhead. The middleware can be implemented on a personal device of the user, such as his smartphone to support this trust. The proof-of-concept can be extended to support additional features such as multiple federations and levels of assurance for attributes and deanonymization. Currently the query is a simple list of attribute identifiers. Adding support for more advanced policy languages could further increase the privacy of the user [55, 52]. Currently, the secure element selects a minimal set of identity providers to satisfy the service provider's request, ensuring optimal performance. However, to increase user control the attribute aggregation module and policies should be extended to allow the user to determine from which identity providers the attributes are aggregated, given the set of possible identity providers.

The secure element application runs on a Mobile Security Card SE 1.0 [8], all other applications run on a DELL E6420 laptop with Intel Core i7-2720QM @ 2.20GHz, 4 cores, 8 logical processors and 8GB RAM. The laptop runs Windows 7 64-bit with Service Pack 1. The proof-of-concept was used to do some performance tests. The presented numbers are the averages and standard

deviations of 20 runs. The establishment of the authenticated session key with providers takes around 1517±85 ms on the secure element and 53±5 ms for the provider. A significant part of the duration on the secure element, around a third, is due to the generation of the ephemeral key pair. A possible strategy to improve performance could be to reuse the same ephemeral key for a number of authentications or have the middleware trigger the card to pre-calculate one or more ephemeral keys when its idle. After the secure channel is established, the secure element can request attributes from identity providers, provide them to service providers and update the `lastValTime`. The secure element operations for fetching attributes from an identity provider take around 300-400 ms, depending on the number of requested attributes. The secure element operations for providing attributes to the service provider take 153±33 ms for only the pseudonym and 211±34 ms for all nine attributes, assuming they are cached on the card. Updating the secure element with a new `lastValTime` takes around 300±37 ms. The operations executed by the provider take less than 5±3 ms for each of the three protocols.

The low-bandwidth communication with the secure element and the computational constraints of the secure element are the major constraints for the performance of the system. The caching strategy has a critical impact on performance as each identity provider that needs to be contacted during authentication greatly increases the duration of the authentication. Caching a set of frequently used attributes on the secure element should ensure that no identity providers need to be contacted for the majority of authentications. The user can tune the caching strategy by marking attributes as *persistent* and having the secure element prefetch attributes from identity providers. Releasing more attributes slightly decreases the performance as more attributes need to be encrypted by and transferred from the secure element to the middleware. Attributes should be relatively short due to the computational and communication constraints of the secure element. To support larger attributes, identity providers could provide the cryptographic hash of the attribute value to the secure element and transfer the actual attribute to the service provider via the middleware. The middleware should also periodically (re)validate the secure element to ensure that this operation does not have to be performed during authentication. This is feasible if the secure element is permanently plugged in a personal device of the user, as discussed in Section 6.2.1.

6.2 Validation

This section validates the identity management system presented in the previous chapter by applying it in two case studies. This research is joint work

with Faysal Boukayoua. This section summarises the most important results relevant to this text. More details can be found in [42, 41]. This section consists of two parts. The first section applies the IdM system to control access to a Web based service provider. The second part tackles interoperability by illustrating how it can be integrated in an already deployed Shibboleth environment.

6.2.1 Out-of-Band Web Authentication Using a Smartphone

Many service providers make their remote services available via the Web. The user browses to the Web page of the service provider where access to the service can be requested. The identity management architecture is used to assert the required information for the service provider to make an informed access control decision. This section validates the identity management architecture by illustrating how it can be used in a Web based context. A prototype is implemented to illustrate the feasibility of our approach.

The user mostly uses his own workstation to consume the services of Web based service providers. However, on some occasions the user might also need to use public workstations or workstation shared by multiple users. On these workstations the user might not have the required privileges to install software. Moreover, malware might be running on the workstation that can compromise the middleware and intercept the PIN of the user. Hence, our system should not require the installation of middleware on the workstation and the user should not have to rely on the workstation to securely handle his PIN and correctly inform the user about the information that will be released.

Design

The user (U) has a mobile device (MD) containing a secure element that implements the SE functionality of the identity management architecture. The smartphone contains an application implementing the IdM middleware functionality. The user requests access to a remote service (SP) via the Web browser (WB) on a workstation.

Figure 6.3 presents the messages exchanged between the different actors to authenticate the user towards the service provider. First, the user requests access to a remote service via the browser on his workstation (1). The service providers requires the user to prove certain personal attributes before granting access. The service provider, therefore, returns a webpage containing a QR code with the authentication challenge (i.e. $cert_{sp}$, `accValTime`, $pk_{eph,sp}$ and $query$) and an authentication identifier (i.e. authID) (2). The authentication identifier

is generated by the service provider and linked to the browsing session of the user. The user now scans the QR code with his mobile phone (3). The mobile phone now informs the user about the pending authentication (i.e. information about the service provider and the attributes that will be released) (4). If the user agrees with the authentication, he transfers his PIN and a possible modified attributes query to the mobile device (5). The mobile device now interacts with the secure element contained in the mobile to generate the authentication response (i.e. $pk_{eph,se}$, $cert_{Co}$ and $\{\texttt{accValTime}||atts\}_{K_s}$). The possible interactions with identity providers are omitted for reasons of clarity. The authentication response is then transferred from the mobile to the service provider together with authID via the xG connection of the mobile phone (6). The service provider can now construct the authenticated session key and use it to decrypt the attributes. The service provider then assigns privileges to the browser session of the user based on the obtained attributes. The browser session of the user is uniquely identified by authID. If the access control policies are satisfied, the user is granted access to the requested service (7).

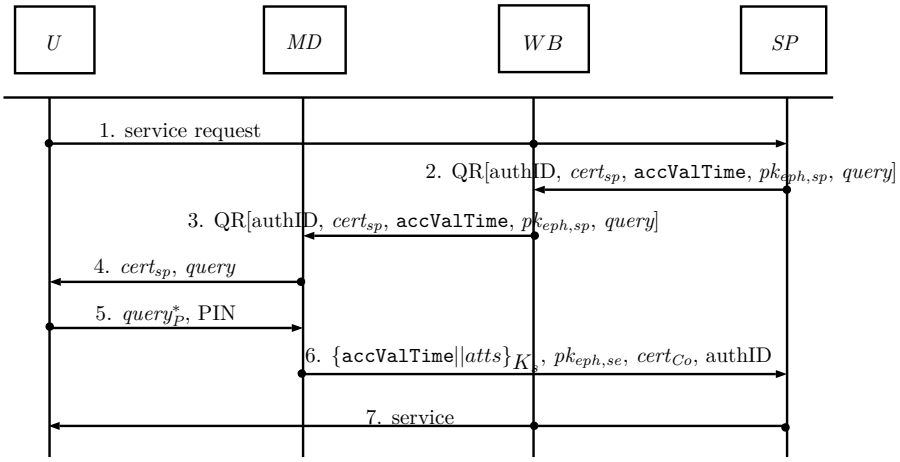

Figure 6.3: Authentication to a Web-based service provider (*U*: User, *MD*: Mobile Device, *WB*: Web Browser, *SP*: Service Provider).

Prototype

This section shortly discusses the software components that were developed for the prototype, namely a smartphone application and the authentication and access control logic for the service provider. The software for the identity providers and the secure element remain the same as in the proof-of-concept

implementation. The hardware used for the realization of the different entities is shown in Table 6.1.

Table 6.1: The hardware platforms for the realization of the different entities (*SE*: Secure Element, *MD*: Mobile Device, *WB*: Web Browser, *SP*: Service Provider).

Entity	**Realization**
SE	Mobile Security Card SE 1.0 [8]
MD	Samsung i9000 Galaxy S: 1GHz ARM Cortex-A8, 512MB RAM The smartphone runs Android 2.3.3
WB, *SP*	DELL E6420: Intel i7-2720QM @ 2.20GHz, 4 cores, 8GB RAM The laptop runs windows 7 64-bit with service pack 1

Smartphone Application. The mobile application implements the middleware as described in the previous section. The mobile application is developed for an Android phone. Since Android applications are also written in Java, large portions of the proof-of-concept's middleware software can be reused. The main modification that needed to be made was the communication with the Mobile Security Card and the user interface. Instead for relying on the PC/SC API, a library from the seek-for-android [12] project that allows Android applications to interact with the smart card chip of the µSD. The application further uses the zxing [9] library to scan and interpret QR codes. The `URLConnection` class is used to perform an HTTP POST containing the serialized `AuthenticationResponse` to the URL contained in the service provider's certificate.

The mobile application also offers multiple management functions to the user. The application shows the user to which service provider (i.e. URL) the secure element is about to authenticate. Furthermore, the application also displays the user attributes (or properties thereof) that are requested. Attributes may be mandatory or optional. If any optional attributes are requested, the user can select whether or not to disclose them.

Service Provider Application. The service provider is implemented on an Apache Tomcat [1] server. Spring Security [13] is used to enforce its access control policy. A Spring authentication module was added to handle the custom authentication protocol. Memcached [7], an in-memory key-value store, links the session information to the authentication identifier.

Evaluation

Nowadays many communication technologies are supported by the current generation of smartphones. However, most of them require dedicated software or configuration on the workstation in order to allow interaction with the smartphone. Optical communication on the other hand, can easily be used to transfer data from the browser, running on the workstation, to the mobile without needing to install any additional software. The mobile device offers a platform for entering the PIN of the user and controlling the authentication without needing to trust the software running on the workstation.

The current prototype focuses on authentication to Web-based services that are consumed via the workstation. However, there are several other scenarios in which a mobile device can be used to access personalised services. For instance, users might need to prove that they are old enough to buy alcoholic beverages at a vending machine. Other scenarios include controlled access to buildings, loyalty discounts, etc. In these scenarios, the smartphone application may use NFC or Bluetooth to connect to the service provider. Apart from the communication layer, no other modifications are required.

A nearby eavesdropper could scan the QR code shown by the browser. This gives the attacker access to the authentication challenge and the authentication identifier. An attacker could use it for a denial-of-service attack by using the authentication ID to invalidate the user's authentication session with the service provider. The authentication ID is different from the session identifier, preventing the attacker from hijacking the browsing session of the user.

6.2.2 Shibboleth Integration

The previous section illustrated how the IdM system can be used for service providers to enforce access control rules in a Web based context by using the software modules developed for the proof-of-concept. Each actor is assumed to install the custom IdM modules. However, in some cases, actors already have a legacy infrastructure deployed and are reluctant to extend/replace it. This section discusses the case in which the service providers use the Shibboleth framework to authenticate its users. This integration allows the service provider to support a wider range of users and increases the application domain for the new IdM system. The latter can play an important role when introducing a new IdM system, since the number of available services is an important factor for users to start using the system.

Design

The user requests access to a remote service via the browser on a workstation. The service provider uses the Shibboleth authentication infrastructure to authenticate its users. Hence, when the user requests access to a protected resource of the service provider, the user's browser is redirected to the Shibboleth Discovery Service. The Shibboleth Discovery Service allows the user to select a trusted Shibboleth identity provider that is responsible for the endorsement of the user's attributes to the service provider. To avoid naming confusion with the identity providers from the IdM system presented in the previous chapter, the Shibboleth identity provider is referred to as identity broker.

In a standard Shibboleth deployment, the user authenticates to the identity broker allowing the identity broker to retrieve the user's information from a database. However, in our design, the identity broker no longer stores the attributes of the user. Instead, the user proves the attributes during authentication towards the identity broker. Attribute storage and provisioning are moved to the IdM architecture. Each Shibboleth federation appoints one identity broker that holds a separate certificate for each service provider. Each certificate contains the information and rights of a specific service provider to obtain a set of attributes from a secure element. The task of the identity broker is to convert the attributes retrieved during authentication to a Shibboleth assertion that can be interpreted by the service provider.

Prototype

The prototype builds upon the open source implementation of Shibboleth v2. A standard service provider is deployed. Some modifications were made to the identity provider software to support the custom authentication protocol. The prototype is tested on the same hardware as in the previous case study

For authentication towards the identity broker, the prototype uses the system presented in Section 6.2.1. No modifications are required to the smartphone and identity provider applications. During authentication, the required attributes are provided to the identity broker. The user is informed regarding the released attributes by the mobile device. Since the identity broker uses a unique certificate for each service provider, the user is also informed about the service provider to which he is authenticating. This ensures that the identity broker does not use a certificate from another service provider to be able to request more or a different set of attributes from the secure element.

Support for the custom authentication system is added to the identity broker by implementing a Shibboleth `LoginHandler`. This component uses the service provider logic from the prototype presented in Section 6.2.1 to authenticate the user and obtain the required attributes. The certificate used during authentication depends on the service provider that issued the SAML request.

Evaluation

Next to interoperability with a Shibboleth service provider, the presented approach also provides several advantages compared to a standard Shibboleth deployment. Since the identity broker uses separate certificates for each service provider, the secure element generates unlinkable service provider specific pseudonyms. This prevents the identity broker from linking authentication sessions of the same users to different service providers if no other uniquely identifiable information is released. Moreover, the user is informed about the pending authentication and can selectively disclose his attributes via his smartphone. The required attributes can be aggregated from multiple identity providers. The identity broker is trusted by the user not to abuse the data obtained during user authentication by, for instance, trying to link different pseudonyms of the user or impersonate a user towards the service provider. The identity broker is trusted by the service providers for the provisioning of the user's attributes. The Shibboleth trust model decreases the required trust in one identity provider by distributing this trust over several identity providers that can each provide a subset of user attributes.

6.3 Discussion

This chapter presented two case studies validating the IdM system presented in the previous chapter. Both case studies rely on a mobile device to provide a secure interface to interact with the secure element and manage the authentication. It provides a mobile and convenient platform to manage the credentials of the user. These mobile device are, moreover, typically managed by the user which makes it more trusted than, for instance, a public workstation. Mobile device operating systems are, however, also vulnerable to infection by malware. To increase the security Trusted Execution Environment (TEE) technologies [189, 90, 191, 206, 105, 177] embedded in several commodity off-the-shelf smartphones could be used to provide a secure interface to the user. Ideally, the TEE hardware support ensures that the secure element can only be accessed by a trusted TEE application.

Several identity management systems use SAML tokens to request and assert properties about users. Our implementation, however, uses customized protocols and message formats since parsing SAML messages would greatly increase the complexity and impact the performance [154] of the secure element module. An alternative would be to exchange SAML messages between service and identity providers via the secure element. Although this avoids the need to parse SAML messages on the secure element, it introduces multiple other disadvantages. For instance, the privacy advantages compared to current solutions would diminish as the trust would again shift from the secure element to the identity providers. Hence, the communication overhead to the secure element would significantly increase, which would seriously impact performance. The previous section discussed another approach in which a trusted third party performs a protocol conversion between two IdM systems. Another approach to stimulate adoption is adding support to identity management frameworks such as the Higgins [4] framework. This strategy is not pursued in this text.

6.4 Conclusion

This chapter presented a proof-of-concept implementation of the identity management system presented in the previous chapter. The proof-of-concept consists of several reusable software components that can be used to integrate the IdM system in applications. Several implementation aspects that need to be considered when implementing the IdM system are discussed. The proof-of-concept uses a standardized elliptic curve based authenticated key agreement protocol to provide strong security while maintaining good performance on the resource constrained secure element.

The second part of this chapter validates the identity management system by applying it in two case studies. The first case study handles user authentication towards a Web-based service provider. The user accesses the service via a browser on his workstation. The smartphone of the user is used to authenticate the user towards the remote service provider. A Mobile Security Card is plugged in the smartphone on which the secure element functionality is implemented. The middleware is implemented on the smartphones and allows the user to manage and approve authentication requests. This approach ensures that the user does not need to enter his PIN on an untrusted workstation, reducing the trust required in the workstation. Moreover, the solution does not require the installation of any dedicated software on the workstation, supporting user authentication on workstations on which the user has no administrator privileges. The second case study presents an approach for using the IdM system to authenticate towards a Shibboleth service provider.

This allows the system to be used with legacy infrastructures, increasing the application domain. The approach also provides several advantages compared to a standard Shibboleth deployment. It adds support for attribute aggregation and increases the privacy and control of the user over the disclosure of his attributes. It however, breaks the trust model of Shibboleth and requires the service provider to trust a single identity provider with the provisioning off all the user's attributes. A similar approach can be used for other identity management systems such as OpenID [167].

Chapter 7

An Architecture for TPM-Backed Identity Management

Many identity management systems rely on the workstation to provide an environment on which the user can securely use his credentials, and assess and approve transactions or authentication requests. However, the regular operating systems, such as Windows, Linux distributions and OS X, typically running on these workstations are vulnerable to malware due to the large Trusted Computing Base (TCB) [170]. Using credentials on these workstations, hence, potentially exposes them to abuse by malware [120, 175].

Existing works [142, 141, 140, 48] describe the realization of a Trusted Execution Environment (TEE) on a workstation and present a framework for developing applications running in the TEE. The system works on commodity off-the-shelf workstations and laptops, and does not require the user to buy additional hardware. The trusted execution environment runs next to and isolated from the regular operating system. The system, hence, does not hinder the user in the day-to-day work on his computer but is only activated when user authentication is required. By migrating all credential related operations from the regular operating system to a trusted application running in the TEE, malware is prevented from obtaining the credentials or misleading the user about the pending authentication (i.e. secure I/O). To realize these features, the user must be able to distinguish between a trusted application running on the workstation or malware trying to mislead the user.

This chapter introduces the state of the art regarding TPM-based TEEs. Further, an extended enrollment phase is presented that allows the user to establish trust in the software running in the trusted execution environment on the workstation via his smartphone. This ensures that the user is correctly informed regarding the pending transaction. During the enrollment phase, the mobile device also obtains the public key of the trusted application. This allows secure data transfer between the mobile device and the TEE application. An USB-UHCI driver is added to an existing framework for developing TEE applications. This allows TEE applications to access external hardware security technologies via the USB interface. The advantage of this setup compared to trusting the mobile phone to handle the user's credentials is that the TEE can interface with hardware technologies such as a smart card reader or a biometric scanner whereas mobile phones typically cannot. Moreover, the attestation capabilities of the TPM can be used to assure a relying party that the correct protocols were executed. This setup is used in Chapter 8 where two advanced case studies are presented that use the TEE environment to increase the security and privacy of identity management systems.

7.1 Preliminaries

7.1.1 TCG Trusted Computing

Nowadays, commodity computers are equipped with a trusted platform module [186]. This is a hardware module physically attached to the computer's motherboard, extending the system with a set of security related features. One of these features is the measurement of the state of the system. To this end, the TPM contains several Platform Configuration Registers (PCRs). These are cleared upon power up and can only be modified using the extend operation, performed inside the TPM. This operation requires two parameters: a PCR register and a value. This value is typically the hash of a binary software component. The result of this operation is a new PCR value, being the hash of the value currently contained in the register and the *value* to extend (i.e. $PCR_n := SHA1(PCR_n \mid\mid value)$).

A transitive trust model is employed: each software component, starting from the Core Root of Trust for Measurement (CRTM) in the BIOS, is responsible for measuring the following component in the chain before passing control. Hence, before loading subsequent software components, the preceding component hashes the binaries of the components to be loaded and extends the result in a specific PCR. As a result, the PCRs represent the state of the system (i.e. the loaded software configuration). Some PCR registers are designated to contain

measurements of specific software components or settings. For example, the BIOS and the motherboard configuration are measured respectively in PCR registers zero and one. Other PCR registers can be freely used by software developers.

Based on this state, the TPM also supports a number of additional operations. Data can be encrypted with the seal operation and only if the system resides in the state specified during the seal operation can the data be decrypted (unseal). Additionally, the quote command returns a proof of the state (i.e. a *quote*) which a, possibly remote, third party can verify (verifyQuote) asserting that the (remote) system runs in a specific (trusted) state. Quote uses the private key and certificate of the TPM (i.e. sk_{tpm} and $cert_{tpm}$) to assert that the operation is performed by a genuine TPM. These credentials can either be generated by the hardware manufacturer/vendor or during an initialization phase. The widely supported TPM specification 1.2 supports both attestation based on an RSA key pair and Direct Anonymous Attestation (DAA) [49]. If the latter is used, the validity of the TPM is asserted without releasing uniquely identifying information.

7.1.2 Trusted Execution Environment

While TPMs have been built-in in workstations for several years now, a more recent evolution is the adoption of TEE technologies such as Intel's Trusted Execution Technology (TXT) [68] and AMD's Secure Virtual Machine (SVM) [80]. These technologies allow the execution of measured code independently of previously executed software. The TPM specification has been extended with additional capabilities to support these new technologies.

These technologies tackle the main disadvantage of using a chain of trust established from the BIOS to the operating system, namely the large trusted computing base. The AMD SVM and Intel TXT technologies were developed to provide a hardware-based Dynamic Root of Trust Measurement (DRTM). A special command is implemented in the processor that takes as input a physical start address referencing code contained in memory. Before the processor begins execution of the referenced code, the code is extended in special purpose registers in the TPM. Where other registers can only be reset by rebooting the workstation, these registers are reset by the previously mentioned command, before the code is extended in the registers. Hardware protections such as disabling interrupts, preventing DMA access to the code in memory and even prohibiting access by hardware debuggers attached to the motherboard, ensure that the measured code is the code executed by the processor. This

technique allows trustworthy assertion of code without requiring an entire chain of trust [123] starting from the BIOS.

McCune et al. [142, 141, 140] presented a framework called *Flicker* that uses these technologies to allow developers to isolate security critical code from applications and run it in a secure environment. The main, possibly untrusted, OS is temporarily suspended after which the sensitive Piece of Application Logic (*PAL*) is securely executed. When the execution of the sensitive code is completed, the OS resumes execution. Typically, the *PAL* extends its state in the TPM with a fixed known value before releasing control back to the OS. This prevents the OS from gaining access (i.e. unseal) to secrets from the *PAL*. Before extending its state in the TPM with a fixed value, the *PAL* can also extend its state with a data entry. The OS can then use the quote command of the TPM to assert that the specified data entry was produced by a trusted *PAL* (i.e. the asserted state equals the known state of the *PAL* extended with, subsequently, the data entry and the known fixed value). The framework supports data transfer between the main OS and the *PAL*. The TPM operations can be used to assert to a remote party that certain data was generated by a trusted *PAL*. The framework supports both Intel TXT and AMD's SVM technology on Windows and Linux based systems. Brasser et al. [48] extended this framework to allow secure user interaction (i.e. input via the keyboard and output via the monitor). An enrollment procedure is used during which a user-specific picture is sealed to the state of the trusted application. This is done in a trusted enrollment environment (e.g. on a freshly installed workstation, not yet connected to the Internet) so that an attacker cannot obtain the picture of the user. Since only that specific application can access the picture, the user is assured that the trusted application is running if the correct picture is shown.

7.2 Related Work

Even though the solution presented above minimizes the trusted computing base, there is no guarantee that even with a small TCB the code correctly represents the desired functionality and that there exist no bugs that can be exploited. Therefore, tools have been developed to check a codebase for the existence of bugs. For instance, the seL4 microkernel, consisting of 8,700 lines of C code and 600 lines of assembler, has been formally proven to be consistent with its specification and free from programmer-induced implementation errors [124]. The proof is constructed and checked in Isabelle/HOL [155], a proof assistant for higher-order logic. Other tools for formally verifying certain correctness properties of C programs are VeriFast [117], Why3 [94]

and Dafny [133]. Similarly, compilers need to be verified [135] that the security properties that are proven on the source code also hold for the compiled code.

During the development of this PhD, the developers of Flicker continued to work on the topic. Subsequent work took a slightly different approach, trying to overcome some of the disadvantages of the Flicker system. One of the disadvantages of Flicker is the intensive use of TPM operations which significantly decreases performance due to the constrained nature of the TPM. In [190], Vasudevan et al. present an eXtensible and Modular Hypervisor Framework (XMHF). The regular OS runs on top of a small hypervisor. The hypervisor leverages hardware virtualization primitives to allow the regular OS direct access to all performance-critical system devices and device interrupts while maintaining memory integrity. This model greatly decreases the complexity of the hypervisor (6018 lines of code) and enables high performance due to the low hypervisor overhead. The hypervisor bootloader uses the DRTM capabilities of modern processor to measure the integrity of the hypervisor. A virtualized TPM [139] running as a *hypapp* on top of the hypervisor and next to the regular OS is used to greatly improve the performance of *PAL*s. This work can, hence, be used to increase the performance of the *PAL* prototypes developed with the Flicker framework further in this text. In [212] this work is extended with a module that enables users to establish trust in the software running on the workstation. A dedicated hardware module with a USB interface is used to interface with the workstation. The hardware module indicates with a red and green led when the system is in a respectively untrusted or trusted state. A similar approach is adopted in [179, 180].

7.3 Design

This section presents an extension of the enrollment phase proposed by Brasser et al. [48]. A mobile device is used to verify the trustworthiness of the *PAL*. The extended enrollment procedure allows the user to establish trust in a *PAL* running on a workstation that is (possibly) infected by malware or untrusted software (e.g. a workstation in a hotel, library). This increases the mobility of the user as he is no longer restricted to his own workstation(s). The enrollment also facilitates secure data transfer between the mobile device of the user and the *PAL*.

7.3.1 Roles

We assume a user (U) carrying a mobile device (MD) and working on a workstation that runs a legacy operating system (W) and supports TEE technologies for running a trusted application (PAL).

7.3.2 Requirements and Adversary Model

Functional requirements

F_1 The enrollment allows the user to establish trust in a PAL running on an untrusted workstation.

F_2 The enrollment phase facilitates secure transfer of data from the mobile device to the PAL on the workstation.

Adversary Model

We assume an attacker that can manipulate the user's operating system and application. Regarding the secure execution environment, the same assumptions as those of the Trusted Computing Group [186] are made. The system is mainly focused on protecting the user from software attacks. Hardware (key)loggers and shoulder surfing are considered out of scope. With respect to the cryptographic capabilities of the attacker, we follow the Dolev-Yao attacker model [85]: attackers cannot break cryptographic primitives, but they can perform protocol-level attacks. The mobile device is trusted by the user.

7.3.3 General Approach

The mobile device verifies the integrity of the PAL running on the workstation on behalf of the user. Subsequently, the mobile device allows the user to select an authentication picture and binds the selected authentication image to the trusted PAL on the workstation. Since only a correct PAL can access the authentication picture, the user is assured that the trusted PAL is running on the workstation if the correct authentication picture is shown on the monitor.

7.3.4 Protocols

Prerequisites

The TEE technologies on the workstation are enabled in the BIOS and the TPM has been certified (i.e. either sk_{tpm} and $cert_{tpm}$ are generated or a DAA credential $cred_{daa}$ is issued). The *PAL* running on the workstation of the user has been initialized. During initialization the *PAL* generates a keypair (pk_{pal} and sk_{pal}) and *seals* it to its state resulting in the sealed object denoted as $keyStore$. The mobile device obtained the certified PCR state of the trusted *PAL*. Hence, it can verify that the application running on the workstation is indeed the intended trusted application. A trusted third party certified the state of the *PAL* after reviewing the source code. The *PAL* could also be open source so that independent developers can verify that the certified state correctly represents the desired functionality.

The mobile required the user to choose a unique authentication picture (img_u) that will allow the user to visually verify that the software running on the workstation is indeed trustworthy.

Enrollment

The mobile device verifies that a valid trusted application is running on the workstation. As part of this protocol, the mobile retrieves the *PAL*'s public key (pk_{pal}), which is used to encrypt data addressed to the *PAL*. The mobile device stores this public key for future authentications. We denote this protocol as the *enrollment protocol*.

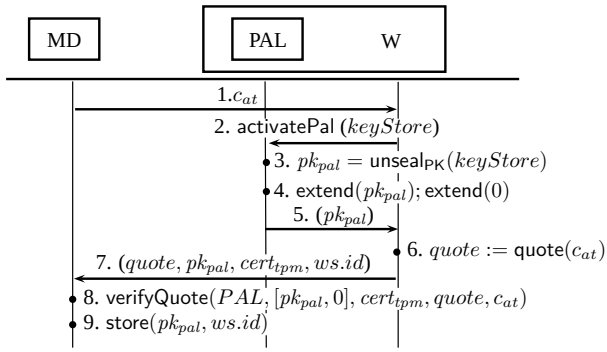

Figure 7.1: The enrollment protocol (*MD*: Mobile Device, *PAL*: Piece of Application Logic, *W*: Workstation).

Figure 7.1 illustrates the steps of the enrollment protocol. First, the mobile sends a random attestation challenge to the workstation (1). The workstation activates the trusted *PAL* with the sealed *keystore* (2) and the *PAL* retrieves its public key from the sealed *keystore* (3). To allow attestation that this public key is indeed managed by the trusted application, the state is extended with this key (4). The *PAL* returns its public key to the workstation which resumes its execution (5). A quote operation on the state resulting from the *PAL* execution is performed using the attestation challenge (6). As the public key was extended in the state, this ensures the authenticity of the public key sent to the mobile. The challenge ensures freshness of the quote. The quote, public key, certificate and an identifier of the workstation is sent back to the mobile, which then verifies the quote (7-8). If the quote verification was successful, the public key is stored together with the workstation's identifier (9).

7.4 Realization

Table 7.1 shows the hardware used for the realization of the different entities. For the functionality of the mobile device, an Android app has been developed. It allows the user to select an authentication picture and, during enrollment, binds the picture to the state of the *PAL*. A subset of the PolarSSL [11] library was used to realize the cryptographic operations in the *PAL*, namely generating and using a keypair. The TPM is used to generate a seed for the key generation. Further, an USB-UHCI stack was added to the Flicker framework to enable access to USB security devices. On the workstation, the TrouSerS[1] TCG software stack is used to implement the attestation of the *PAL* towards the mobile device.

Our *PAL* application consists of two main components. The USB-UHCI stack (1001 lines of code) and the implementation of the enrollment protocol itself consists of 1040 lines of code. This adds a total of 2041 lines of code to the Flicker framework, preserving a minimal trusted computing base.

Table 7.2 illustrates the performance of the prototype. The computations on the mobile have negligible impact on the overall system performance. The *PAL* and the *W* introduce the largest overhead as they use the constrained resources of the TPM. The workstation uses the TPM for the quote operation, the *PAL* for the unseal functionality and the establishment of the TEE.

[1] TrouSerS is an open source implementation of the TCG software stack, more information can be found on `http://sourceforge.net/projects/trousers`.

Table 7.1: The hardware platforms for the realization of the different entities (*MD*: Mobile Device, *PAL*: Piece of Application Logic, *W*: Workstation).

Entity	Realization
MD	Samsung i9000 Galaxy S: 1GHz ARM Cortex-A8, 512MB RAM The smartphone runs Android 2.3.3
PAL, *W*	DELL E4200: Intel Core2 Duo U9400 @ 1.4GHz, 4GB RAM The laptop runs Ubuntu 12.04 32-bit, 3.2.0 kernel

Table 7.2: The performance (i.e. averages and standard deviations of 30 runs) of the enrollment in milliseconds (*PAL*: Piece of Application Logic, *W*: Workstation, *MD*: Mobile Device).

Performance	PAL	W	MD
enrollment	1917±123	725±53	31±4

7.5 Evaluation

7.5.1 Requirements Review

The user relies on his mobile device to correctly verify the integrity of the *PAL* running on the workstation before sealing the authentication picture to the state of the *PAL*. Since, apart from the mobile device, only the *PAL* can access the authentication picture, the user is assured that the trusted *PAL* is running if the image is shown on the monitor, satisfying requirement F_1.

During the enrollment procedure, the mobile device obtains the public key of the *PAL*. Since only that *PAL* can access the corresponding private key, the public key can be used to securely transfer data between the mobile device and the *PAL*, satisfying requirement F_2.

7.5.2 Security and Privacy Considerations

The main focus of the system is protecting the user from software attacks on the workstation, as these are the most common and scalable types of attacks. Although TPMs are not designed to be secure against hardware attacks, they are not trivial to execute for regular users. Since a TPM's credentials can be revoked, compromising a TPM does not affect the security of the entire system.

During the enrollment procedure, malware running on the workstation could relay the data from the mobile device to another workstation on which a trusted

PAL is running, effectively enrolling the mobile with a different workstation. When the user wants to start the *PAL* after finishing the enrollment procedure, the malware on the workstation of the user could start the *PAL* running on the other workstation and, subsequently, capture the image shown on the monitor via, for instance, a camera. This allows the malware on the workstation to show the correct image and, hence, mislead the user into believing that the trusted *PAL* is running on that workstation, while it is in fact running on another workstation. This may lead towards the user entering security sensitive information on the workstation infected with malware. This relay attack is described in [161] and referred to as the Cuckoo attack. The system is, hence, more suited to use on private workstations where this attack is much less pertinent than in a public setting. To extend the system to public workstations, additional measures can be implemented to protect against this attack. Several approaches have been proposed [161, 143, 97] to tackle this problem. A QR-code containing a hash of the TPM's public attestation key could be embedded in the workstation's case. The smartphone can then scan the QR-code and verify if the TPM used to generate the attestation is embedded in that workstation. A second approach is to trust the machine's BIOS to correctly transfer the TPM's public attestation key to the smartphone. The user reboots the machine and enters the BIOS to obtain the TPM's public key, before executing the enrollment phase. This assumes that the BIOS is protected from malicious updates (e.g. by only installing signed updates).

The system relies on the mobile device to correctly verify the *PAL* on the workstation and securely manage the authentication picture. The mobile device is typically owned and managed by the user, making it more trustworthy than workstations not managed by the user. Moreover, current mobile device operating systems have, by default, stronger application isolation properties than commodity workstations and the market based software management system is more closed than their desktop counterparts. This hinders the development and deployment of malware on these devices. However, these operating systems also rely on a huge trusted computing base making the existence of exploitable bugs inevitable. Moreover, some users change the default security settings (e.g. enabling side-loading, rooting and unlocking the bootloader) to gain more control over their device but making it more vulnerable to malware. The impact of malware on the operating system could be mitigated by using recent technologies that focus on establishing a trusted execution environment on mobile devices [191, 206, 105, 177] to verify the attestation of the workstation and seal the authentication picture to the state of the *PAL* in an isolated environment.

The system assumes that the authentication picture of the user is only accessible to the *PAL*. While this assumption holds under the TPM security

assumptions for malware running on the OS, during the authentication of the *PAL* towards the user the authentication picture is shown on the monitor. This allows an attacker to capture the image by taking a picture from the monitor and misleading the user in subsequent authentications. The current pairing protocol, hence, may be secure on the workstation of the user but for public areas and shared workstations, a minor modification of the enrollment procedure is required to ensure adequate security in those settings. Instead of sealing the authentication picture to the state of the *PAL* and using that image in all subsequent authentications on that device, the enrollment procedure should be executed each time the users wants to use the secure environment on that system. During enrollment, the mobile device chooses a random authentication picture from a large set of pictures and securely transfers it to the *PAL*. The user can compare the image on the monitor and mobile device to verify the integrity of the *PAL*. This decreases performance as the enrollment procedure is executed more often, but increases the security for public workstations. If the home workstation is shared by multiple users, multi-user support could be added by requiring the user to enter a PIN before the authentication image is shown. Another approach is used in [192]. A dedicated embedded device with a USB interface is plugged into the workstation. The embedded device can verify the attestation of the workstation. If the workstation is running a specific trusted application, a light attached to the embedded device turns green. This replaces the procedure with the mobile device and authentication picture and, hence, prevents attacks relying on the authentication picture. Moreover, since the functionality of the embedded device is limited, it is less vulnerable to malware attacks compared to the mobile device. It, however, increase the cost and requires the user to carry around an additional device.

Recent work by Wojtczuk et al. [207, 208] also manages to compromise the rather novel SEE implementations via security bugs in BIOS and PC firmware. They illustrate that TPM-based TEE implementations still require secure BIOS implementations. This can be verified using the respective TPM PCRs representing the BIOS binary. These vulnerabilities are not specific to our solution and should be remedied by the platform manufacturer. These vulnerabilities are, moreover, rare and typically very hard to exploit compared to common Trojan horse or phishing attacks. Overall, using the TEE can increase the security of existing identity management systems.

7.5.3 Discussion

While the TPM and related technologies can be used to provide several interesting security features, the adoption of these technologies remains very

limited. This section discusses several problems that hinder wide adoption.

For the service provider to trust the attestation of a TPM, he must be assured that the attestation was generated by a genuine TPM. The TPM should, therefore, be equipped with a certificate from the TPM or platform vendor. Many vendors, however, ship their products with uncertified TPMs. Before users can start using the *quote* functionality of their TPM, a third party should, hence, be contacted to certify the TPM. In a closed environment such as a corporate environment, a central authority (e.g. the IT department) can be made responsible for the TPM certification process. Each entity in the environment trusts the certificates issued by that authority. In open environments, such as citizens using TPM attestation functionality towards third parties, the TPM certification is less scalable since it is not a procedure that can be executed remotely. This hinders wide adoption of TPM based systems that use the attestation capabilities of the TPM. The most scalable solution would be to have the TPM/platform vendors to start certifying the TPMs in their devices.

The TPM and TXT/SVM technologies need to be activated in the BIOS before they can be used. Where accessing and changing BIOS settings is not an everyday task for most computer users, it might even be an insurmountable task for some computer users. Settings in which a central authority is responsible for managing the hardware (e.g. the corporate application domain) could be an interesting first validation path.

The workstation can attest towards a relying party that a certain result was produced by a specific application. This gives a relying party assurance that the data was generated following specific protocols. The application and its functionality is uniquely identified/represented via its PCR values. In order to provide meaningful attestation, the relying party should be able to check that the application correctly implements the desired functionality. Having each relying party go through the source code, however, is not a scalable approach and requires considerable expertise from the relying party. Hence, this process should be delegated to a trusted third party which can bind PCR values with an informal description of the associated functionality and a serial number. This trusted third party can revoke/blacklist specific versions of a *PAL* if a bug is discovered by adding the serial number to the revocation list. During the attestation process, the relying party can verify the revocation status of the *PAL* by checking that the serial number is not contained in the revocation list composed by the trusted third party. Ideally, the source code of the *PAL* is also open source so that independent developers can also check if a certain PCR state matches the desired functionality. Currently, there is no real infrastructure in place to realize this *PAL* certification process. To limit implementation flaws that could be exploited, the functionality of the *PAL*

should be kept to the minimum. A small TCB decreases the chance of bugs and suggests that formal verification is possible.

7.6 Conclusion

In this chapter we extended a system for realizing a trusted execution environment on commodity off-the-shelf workstations or laptops. The user can establish trust in a software component running in the trusted execution environment using his mobile device. Subsequently, the user can visually verify that the trusted application is running on and in control of the workstation. The trust can be established on workstations possible infected by malware.

The trusted execution environment can be used to increase the security and privacy of existing identity management systems by executing security sensitive operations in the trusted environment. Moreover, the attestation capabilities of the TPM can be used to assert to a relying party that certain operations were executed in a secure execution environment. This allows service providers to enforce a minimal security level.

Chapter 8

TPM-Backed Identity Management: Validation and Case Studies

The previous chapter described a system for establishing trust in an application running in a trusted execution environment on the workstation of the user. During the enrollment procedure, a mobile phone was used to verify the integrity of the application running in the trusted environment. The user selected an authentication image that is bound to the trusted application. The user is assured that, when that specific image is shown on the monitor, a specific trusted application is running as only this trusted application has access to the image. During enrollment, the mobile phone also retrieved the public key of the trusted application, allowing secure transfer of data from the mobile device to the trusted application. The trusted execution environment on the workstation provides ample opportunities to increase the security and privacy in existing identity management systems. This chapter explores these opportunities by presenting two complementary case studies that each tackle several security and privacy issues related to identity management systems/technologies. To demonstrate the feasibility of our approach, a prototype was implemented for both case studies.

1. Traditionally, a user requires substantial trust in workstations for correctly handling his credentials (e.g. password/login). Unfortunately, malware and compromised software makes them unsuitable for secure credential management. Credentials are easily stolen and the user cannot

trust what is being displayed on his workstation, obstructing informed consent. Further, service providers trust the user not to share his credentials with other users.

The first case study, *client-side biometric verification*, presents a new solution that addresses these issues. Credentials are bound to the owner using biometrics, effectively impeding abuse through credential sharing and theft. Binding the user's biometrics to an X.509 certificate can, for instance, be achieved by including a cryptographic hash of the biometric template in the certificate. The biometric verification is performed on the client side, preserving the privacy of the user. The solution ensures that the user is correctly informed about the pending authentication, preventing abuse by malware.

2. Smart cards are popular devices for storing authentication credentials because they are easily (trans)portable and offer a secure way for storing these credentials. They have, however, a few disadvantages. First, most smart cards do not have a user interface. Hence, if the smart card requires a PIN, users typically have to enter it via an (untrusted) terminal. Some smart cards contain a limited user interface (i.e. a PIN pad with a small display). This interface is used as an alternative to the regular contact/contactless interface. The user manually transfers the challenge from the terminal to the smart card, confirms the transaction with his PIN and, subsequently, enters the response generated by the card in the terminal. However, if larger amounts of data (e.g. certificates, identity information) need to be transferred from the card to the terminal, this is not a feasible solution. Alternatively, a smart card reader with PIN pad and display can be used. This is a convenient solution if such a reader is available. However, currently far from most smart card readers have an embedded PIN pad. For instance, some laptops have a smart card reader built-in. These require the user to enter his PIN via the keyboard. Second, smart cards are resource constrained devices which impedes the adoption of advanced privacy-enhancing technologies (PETs) such as anonymous credentials or, at least, they require quite a lot of assistance (collaboration) from that same untrusted workstation.

   The second case study, *increasing security and privacy in eID authentication*, presents a new solution that addresses these issues. It allows users to securely enter their PIN via the workstation and transfer it to the smart card. The solution further extends existing smart card assisted authentication technology based on X.509 credentials with privacy-preserving features such as multi-show unlinkability and selective disclosure. The system can, hence, be used to improve the privacy properties of these rolled-out infrastructures. It improves the security

of the user interface (i.e. secure input of PIN) and preserves the privacy of the user (i.e. selective disclosure of user attributes).

8.1 Client-Side Biometric Verification

Merely using digital credential technologies is not sufficient to fulfill the complex security and privacy requirements of identity management systems. Credential technologies themselves, for instance, do not prevent users from *sharing* their credentials (e.g. digital credentials can be copied and distributed among users). Moreover, credentials can also be *abused by malicious software*. For instance, malicious software can use the credentials of the user to access personalized services, without consent of the user. This impedes abuse detection and consequently credential revocation.

These issues can be tackled by binding the credentials to the owner. This section presents a new solution for activating credentials bound to the owner by means of biometrics. The verification is performed on the workstation by an application running in a trusted execution environment. It therefore uses the system described in the previous chapter. In the prototype solution the user's credentials are stored on his mobile device and are bound to his biometrics. Both the biometric scan and its binding to the credential are verified in the TEE at the client side, trusted by both the user and the service provider. This strategy avoids leaking biometric information to, for instance, the service provider and requires no additional hardware infrastructure to be rolled out.

The *contribution* of this section is threefold. First, it presents a solution for the secure verification of biometric traits on a workstation by applying TEE technologies. Second, access to remote services is controlled by credentials that can only be used after a successful biometric verification on the workstation. A mobile device is used to carry the user's credentials. Third, a prototype implementation of the system was realized, validating our solution. For the implementation, a biometric scanner driver and algorithm were added to the Flicker framework.

8.1.1 Related Work

Some credential systems such as the Identity Mixer library [58] provide all-or-nothing non-transferability to discourage users from sharing their credentials. All the credentials of the user are tied together. Hence, assuming that the user owns at least one valuable credential, he will not be willing to share

his credentials with other users. A similar approach is PKI-assured non-transferability where the credentials are bound to a valuable secret outside the system (e.g. credit card information). Whereas these systems focus on the discouragement of credential sharing, the system proposed in this section also addresses abuse prevention after theft and informed consent (i.e. the owner of the used credential confirms or aborts the transaction based on the correct transaction information).

Another approach to prevent digital credentials from easily being copied or shared is by embedding them inside tamperproof hardware. The system proposed in [51] leverages the DAA [49] protocol, available in Trusted Platform Modules (TPMs) of modern workstations, to bind the user's anonymous credentials to a TPM. Similarly, smart cards are used to implement anonymous credential systems [35, 178, 201, 34, 150]. Although smart cards prevent the credentials from being copied, they do not fully prevent sharing of the credentials. Moreover, anonymous credential systems have other unsolved issues, such as their performance and correctly informing which information is being disclosed.

Biometry can be used to uniquely identify a person [119, 118]. Commonly used biometric traits include a fingerprint, iris, face and voice. A special purpose sensor device is used to read the biometric trait of the user. During enrollment, a distinguishing feature set is extracted from the biometric data and stored as a *biometric template* (BT_u) of the user. During authentication, the user scans his biometric trait (BioScan) and the resulting feature set is *matched* to the feature set contained in the template of the user. Based on the similarity of the two sets, the authentication is either accepted or rejected (BioVerify).

Biometrics can, hence, be used to bind credentials to the owner. For instance, in [114, 39] the *wallet with observer* architecture [66] is extended to include biometric authentication towards the observer. The user is issued a tamperproof card containing his credential and biometric template. To use the credential in the card, the holder is required to scan his biometric data. Only if the scanned data matches the template stored in the card, the credential is activated. As an example, a privacy preserving identity card has been designed [78, 79] taking advantage of this approach. Another approach [37] uses *fuzzy extractors* [83, 84] to generate a cryptographic key based on the retrieved biometric features. This key is never stored and the tamperproof device is trusted to erase the value after authentication. Hence, fresh biometric readings are required to reconstruct the cryptographic key. These systems focus on the prevention of abuse through theft and sharing, but do not fully realize the aspect of informed consent. This is especially important in case anonymous credentials are used. Moreover, tamperproof devices rarely receive software updates and their resource constrained nature impedes the performance of

complex credential technologies.

8.1.2 Design

This section first lists the different actors in the system, followed by the requirements and a general description of the system. Finally, a detailed description of the protocols is presented.

Roles

We assume a user (U), carrying a mobile device (MD). The mobile device stores the user's credentials and is used as a credential vault for accessing remote services (SP) from the workstation. The workstation runs a legacy operating system (W) and supports TEE technologies for running a trusted application (PAL). A biometric scanner is attached to the workstation.

Requirements

Functional Requirements.

- F_1 The system uses the user's credential to authenticate the user towards the service provider.
- F_2 The system is extensible and modular, allowing for new biometric systems or algorithms and credential technologies to be included.
- F_3 When vulnerabilities are found in the system, software updates are easily applied.

Security and Privacy Requirements.

- S_1 The service provider is assured that the owner of the credential approved the authentication.
- S_2 Malicious software cannot mislead the user into approving malicious signing transactions or authentication attempts.
- P_1 The system protects the biometric information of the user.

General Approach

A user authenticates on the workstation towards a remote service provider. The service provider requires that the user proves ownership of the used credentials, before allowing access to its services. To his end, the user's credentials are bound to his biometrics. Binding the user's biometrics to an X.509 certificate can, for instance, be done by including a cryptographic hash of the biometric template in the certificate. For other types of credentials such as anonymous credentials similar principles can be applied. The verification of the biometric binding is further denoted as verifyBinding. The verification of the biometric binding between the credentials and the user is performed by a dedicated trusted application (i.e. the *PAL*). In addition, the *PAL* informs the user about the details of the pending authentication. This ensures that malware cannot mislead the user into approving malicious transactions. This is especially important in case anonymous credentials are used, as the user should give consent on the selective disclosure of attributes. The *quote* functionality of the TPM is used to assert to service providers that a trusted *PAL* properly executed the verification. The credentials and biometric data are stored on the mobile device of the user and are only released towards the trusted *PAL*, running on the workstation. This ensures that the user does not release his biometric data to malicious applications.

Adversary and Trust Model

The same adversary model as the previous chapter is used. The *PAL* is trusted by both the user and the service provider. The user trusts the *PAL* not to release his biometric information to third parties and for informed consent regarding the pending authentication. The service provider trusts the *PAL* to correctly verify the biometric binding between the user and the used credential(s). This trust is supported by the attestation of the *PAL* towards the service provider and mobile. The state of the trusted *PAL* can be certified by a trusted third party. The *PAL* should also be open source so that independent developers can verify that the certified state correctly represents the desired functionality.

Protocols

Prerequisites. The user has gone through the enrollment phase as described in the previous chapter. Hence, the user knows the trusted *PAL* is running if the correct picture is shown and the mobile device can securely transfer information to the *PAL* using the latter's public key obtained during enrollment. The

service provider obtained the certified PCR state of the trusted *PAL*. Hence, the service provider can verify that the application running on the workstation is indeed the intended trusted application. A credential issuer issues credentials bound to the user's biometrics, which are stored on the user's mobile device.

User Authentication. Figure 8.1 presents the protocol for authenticating the user towards a remote service provider. First, the user requests access to a protected resource from the remote service provider (1). The provider responds with an authentication request containing its certificate, an authentication challenge and an attestation challenge (2). The mobile now receives the authentication challenge, certificate of the service provider and a unique identifier of the workstation (3). The mobile informs the user about the workstation on which the authentication will be performed and towards which service provider (4). If the user acknowledges, the mobile signs the authentication challenge with the user's credential and encrypts the signature, the user's authentication certificate, the biometric template and the user's unique image using the public key of the *PAL* (5-7). The encryption is sent to the workstation where it is then passed as a parameter to the *PAL*, together with the *PAL*'s sealed keys in *keyStore* (8-9). The *PAL* now unseals its private key to decrypt the encrypted data *enc* (10-11). Subsequently, the binding between the biometric template and the authentication credential is verified (12). If the verification succeeds, the user is informed about the pending authentication and requested to scan his biometric data. The user's unique image is shown to indicate to the user that the trusted environment is running (13). The user can, hence, trust all information shown on the monitor. To acknowledge the authentication, the user scans his biometric data using the biometric scanner attached to the workstation (14). As the *PAL* is in complete control of the workstation, it can directly interact with the hardware. This ensures that the data shown on the monitor and read from the biometric scanner cannot be tampered with. The *PAL* can now verify if the biometric template matches with the scanned biometric data (15). Upon successful verification, the *PAL* is assured that the user is the owner of the authentication credentials and *extends* its state with the signature, authentication certificate and the certificate of the service provider (16). The *PAL* ends its execution and returns the user's signature and certificate back to the regular OS, that resumes its operation (17). The OS now performs a *quote* operation on the state resulting from the *PAL* execution (18). This quote attests towards the service provider that the trusted *PAL* indeed verified the biometric binding (i.e. the *PAL* state is extended with the user's certificate and signature) and that the user was shown the correct information about the service provider (i.e. the *PAL* state is extended with the service provider's certificate). The resulting quote is sent to the service provider along with sig, $cert_u$ and $cert_{tpm}$ (19). The service

provider now verifies the quote, the user's certificate and signature (20-21). Upon success, the service provider grants access to the requested resource (22).

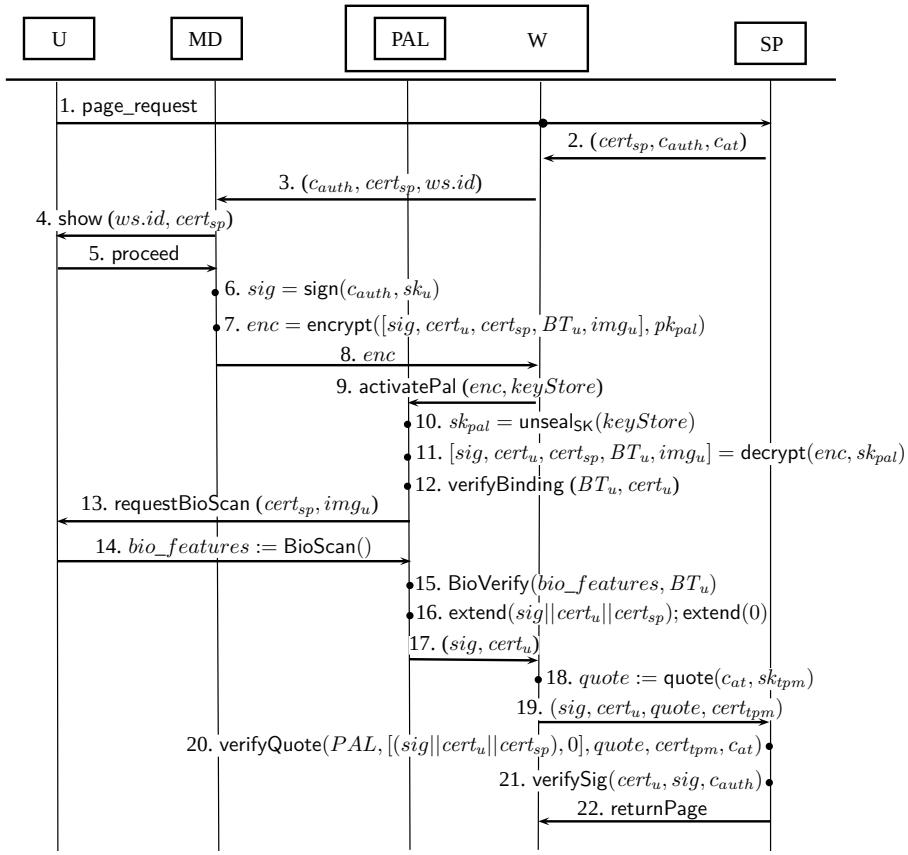

Figure 8.1: The authentication protocol (U: User, MD: Mobile Device, PAL: Piece of Application Logic, W: Workstation, SP; Service Provider).

8.1.3 Realization

For the realization of the prototype, an off-the-shelf USB fingerprint scanner (i.e. Eikon II UPEK [2] fingerprint reader) was used as biometric reader. The user's credential is a X.509 certificate with a 1024-bit RSA key. The user's fingerprint template is bound to his authentication credential by including a cryptographic hash of the fingerprint template in the X.509 certificate. The

same hardware is used as in the previous chapter for the realization of the different entities.

For the functionality of the mobile device, an Android app has been developed. The service provider is implemented on an Apache Tomcat [1] server. Spring Security [13] is used to enforce its access control policy. A Spring authentication module was added to handle the custom authentication protocol. Nevertheless, the focus of the prototype lies in the development of the software components on the workstation. The user accesses the service provider via the browser on the workstation. When authentication is required, the authentication request (*bioauth:authRequestData*) is forwarded to a local application on the workstation. For demonstration purposes a bidirectional network connection is setup between the local application and the mobile device. To this end, a QR code, containing the workstation's IP address, is displayed by the local application and scanned with the mobile device. Using this IP address, the mobile connects to the workstation and uses this channel for further communication. The local application also informs the user about the progress of the authentication protocol. Note that the actual authentication is delegated to a background daemon that accepts incoming TCP/IP connections from the mobile of the user and has the required privileges to suspend the OS and start the *PAL*.

For the biometric verification, a fingerprint driver for the Eikon reader was implemented on top of the USB-UHCI stack. For parsing X.509 certificates and other cryptographic operation, parts of the PolarSSL [11] library were used. When the *PAL* has finished execution, the daemon uses the TrouSerS [14] TCG software stack to implement the *quote* operation which is used to attest the *PAL* towards the service provider.

Our *PAL* application consists of three main components. The Eikon fingerprint driver (447 lines of code) running on top of the USB-UHCI stack (1001 lines of code). The implementation of the *PAL* protocol itself consists of 1040 lines of code. This adds a total of 2488 lines of code to the Flicker framework, preserving a minimal trusted computing base.

Table 8.1 illustrates the performance of the prototype. The computations on the mobile and server have negligible impact on the overall system performance. The *PAL* and the *W* introduce the largest overhead as they use the constrained resources of the TPM. The workstation uses the TPM for the quote operation, the *PAL* for the unseal functionality and the establishment of the SEE. The actual scanning of the fingerprint is not included in the measurement to avoid impact of the user interaction. The measurements of the *PAL*, however, do include the initialization of the USB stack and fingerprint driver. Once, the reader is initialized, the actual swiping is only a fraction of the total time.

Although the authentication phase of the *PAL* is the most time consuming operation, the user experience isn't degraded as some operations can be performed while the user is reviewing the authentication details on the monitor.

Table 8.1: The performance (i.e. averages and standard deviations of 30 runs) of the authentication in milliseconds (*PAL*: Piece of Application Logic, *W*: Workstation, *MD*: Mobile Device, *SP*: Service Provider).

Performance	**PAL**	**W**	**MD**	**SP**
Authentication	4553±153	748±56	42±3	8±0

8.1.4 Evaluation

Requirements Review

Functional Requirements. The mobile device signs the authentication challenge of the service provider with the user's authentication credential. The authentication response is transferred to the service provider over a secure channel, authenticating the user (cf. F_1). The *PAL* can easily be updated by distributing a new binary to the workstation and certifying the new state of the application with the mobile device of the users and the service providers. This can be managed by a trusted third party that certifies and revokes these states. The software on the mobile device and server can be updated using traditional mechanisms. This allows the integration of additional biometric and/or credential technologies and security updates to be installed, satisfying requirements F_2 and F_3.

Security and Privacy Requirements. The authentication credentials of the user are bound to his biometrics. The *PAL* ensures the correct verification of the user's biometrics before asserting the authentication. The service provider can verify the assertion and, hence, check that the actual owner authorized the authentication, satisfying requirement S_1.

The mobile device verifies integrity of the *PAL* running on the workstation, before sending any personal data. Moreover, the user is assured that the trusted application is running, as his personal picture img_u is shown on the workstation. The user is, hence, assured that the provided information about the pending authentication is correct. Furthermore, the *PAL* binds the verification process to the service provider presented to the user. This satisfies security requirement S_2.

In the prototype, the biometric template of the user is bound to the user's certificate by including a cryptographic hash of the template in the certificate. This prevents biometric information of the user from being leaked when the certificate is released. The mobile device only discloses the biometric template to a trusted *PAL* (i.e. the *PAL* does not reveal the fingerprint to a third party) over a secure channel. Moreover, the user can verify that a trusted *PAL* is running when scanning his fingerprint. This prevents freshly scanned biometric information of the user from being leaked, satisfying requirement P_1.

Security and Privacy Considerations

Currently, the system assumes that all service providers require biometric verification. If a service provider does not require this proof of ownership, malware running on the workstation could mislead the user into authenticating to this service provider. If not all service providers require biometric verification, the protocol requires some minor modifications to ensure that an authentication response for a trusted service provider is not abused to authenticate the user towards a service provider that does not require biometric verification. One simple solution could be to have the mobile verify the service provider's certificate and encrypt the user's signature with the contained public key.

Ideally, the biometric reader used for obtaining a biometric scan of the user is bound to the workstation on which the user is working. However, most biometric scanners are plug-and-play and can easily be removed and replaced with other hardware devices. If a fingerprint scan is eavesdropped, it can be replayed as a fresh reading. This risk can be mitigated by implementing a cryptographic protocol between the trusted application and the reader to ensure freshness of the scan. Although hard to achieve, to prevent relay attacks, the *PAL* should also be able to verify that the used biometric reader is actually attached to the workstation on which the *PAL* is running.

The system presented in this section increases the user's privacy with respect to the biometric authentication. Nevertheless, this system is easily extended to further increase privacy by supporting anonymous credentials in which no linkable information should be released to the service provider. In the prototype, the *quote* generation requires a certificate bound to the TPM. This makes all transactions performed on a single workstation linkable. Therefore, modern TPMs also support the *DAA* protocol. This protocol allows anonymous attestation of the platform. As such, the system only leaks the state of the *PAL* and the data disclosed during the user authentication.

The system focuses on protecting the authentication credentials/process from malware on the workstation. However, after the user has established an authenticated channel from the workstation to the service provider, malware could take over the established channel and access the service. The system, hence, does not fully neutralize the impact of malware on the workstation but limits the scope of the abuse to the session established by the user. To increase the security, high value transactions can be processed by the trusted application.

Note that it is even possible to support password based authentication with our system. In that case the certification authority should bind the biometric data to the login or username.

Applicability

This section discusses three possible application domains in which the system presented in this section can be used to realize increased security compared to currently deployed systems, namely *secure corporate networks*, *eID systems* and *online banking*.

Secure corporate networks allow local and remote employees to login with their workstation to the corporate network and access protected resources. Using the biometric authentication system, the authentication server is assured that the correct employee is working on the device. This mitigates the impact of an attacker getting access to a device on which the authentication credentials are stored or company employees sharing their credentials, bypassing the access rights associated to their credential. This is a complementary solution to the Trusted Network Connect (TNC) [185] architecture proposed by the Trusted Computing Group (TCG).

eID systems typically allow the user to access a wide range of personal services. This can go from very privacy sensitive services such as online tax submission to less privacy sensitive services such as pay-per-view news site. While these privacy sensitive services will typically only be accessed from a trusted home computer, other services might be accessed on workstations not fully trusted by the user. However, when using the same authentication credentials for the privacy sensitive and the other services, malware on an untrusted workstation could use the authentication credentials to access other services than requested by the user. The system presented in this section prevents this type of abuse by correctly informing the user about the service which will be accessed and can, hence, be used as an alternative to a smart card reader with a pin pad and a display.

Online banking enables a wide variety of banking services (e.g. viewing the status of your bank accounts, loans and execute bank transactions) via a workstation connected to the Internet. The user owns a credential with which he can log in and authorize transactions (e.g. wire transfer).

In current eBanking systems this secret is stored in an smart card (i.e. the bank card of the user). To authorize transactions, the details of the transaction are transferred to the bank card that subsequently generates an authorization response. This approach protects the credentials of the user but is vulnerable to phishing attacks. To prevent fishing attacks banks can use an offline device in which the user can insert his card to generate Transaction Authentication Numbers (TANs) based on information entered by the user in the device. To increase usability, banks can have users only enter partial information about the transaction (e.g. the amount and the last four numbers of the destination bank account). If only partial transaction information is entered, targeted attacks are still possible. Our approach tackles this since the secure execution environment application correctly informs the user about the pending transaction. It, moreover, replaces PIN authentication of the user with stronger biometric authentication.

Comparison with Existing Systems

As discussed in related work, several systems for misuse protection of credentials exist. For this comparison, the existing systems are categorized as follows. *Hardware based protection (HBP)* systems [51, 35, 201] rely on tamperproof hardware to prevent credentials from being digitally copied and, hence, easily abused. *Software based protection (SBP)* systems [58] rely on binding the credentials of the user to a valuable secret of the user, discouraging users to share their credentials. Finally, *biometry-based protection (BBP)* systems [114, 39, 66, 79, 78, 37] embed the user's credentials in a tamperproof module that requires biometric authentication of the user before the credentials can be used. The results of the comparison are summarized in Table 8.2.

Table 8.2: Comparison between the system proposed in this section and existing misuse protection systems.

	HBP	**SBP**	**BBP**	**Our system**
Informed consent	No	No	No	Yes
HW resources	Constr.	Powerful	Constr.	Powerful
Protection	Copy prev.	Disc. sharing	Bio. bind.	Bio. bind.
Flexibility	Low	High	Low	High

the owner of the used credential confirms or aborts the transaction based on the correct transaction information

Informed consent ensures that the owner of the credential confirms or aborts the transaction based on the correct transaction information. The tamperproof hardware components used in HBP and BBP systems typically do not allow direct communication with the user. Therefore, other, potentially compromised, devices are required to inform the user about the transaction. In SBP systems, the credential operations are typically executed on a regular workstation. In the system presented in this section, the trusted application has full control over the hardware of the workstation and can, therefore, use the monitor to reliably inform the user about the transaction. Note that the BBP system could realize similar features if a smart card reader with a display and biometric scanner is used.

The HBP and BBP both rely on tamperproof hardware components to store the user's credentials and execute the credential and biometric operations. These devices are typically resource constrained limiting the usage of computationally intensive credential technologies such as anonymous credentials and the number of authentication credentials that can be stored inside those devices. The SBP and the system proposed in this section are implemented on a general purpose workstation.

The HBP system implements misuse protection by embedding the credentials in a tamperproof module preventing them from being digitally copied. The SBP system discourages the user from sharing his credentials. The BBP and the approach discussed in this section both implement misuse protection by binding the credentials to a specific user using biometrics. The solution presented in this section, however, could easily be extended to support these credentials without checking the biometric binding.

The HBP and BBP both rely on tamperproof modules for executing credential and biometry related operations. The software installed on these modules is typically difficult to update providing less flexibility compared to the SBP and the system presented in this section. These systems run on general purpose hardware in which updates can be part of the update infrastructure of the operating system.

8.2 Increasing Security and Privacy in eID Authentication

This section presents the second case study of this chapter. Many governments are issuing eID cards that enable citizens to authenticate and prove several personal properties (e.g. name, address, birthdate). This allows the user to establish a secure authenticated session between a workstation and a remote service provider. The remote service provider can control access to his service based on the released information. These identity card systems are often implemented using a smart card to protect the credentials of the user. Typically user consent (e.g. via a PIN) is required before the credentials can be used for authentication.

These systems, however, also have multiple drawbacks. First, as with many smart card based systems, there is no trusted interface with the card [101, 26]. The user typically enters his PIN via the workstation. This allows malware on the workstation to misinform the user about the pending transaction [120, 175] or intercept the PIN which may lead to further abuse. More advanced smart card readers with a PIN pad could tackle this problem. They, however, come at a higher cost than regular card readers and are, hence, a less popular choice. Further, many laptops have built-in smart card readers that also rely on the laptop for user interaction. Second, many systems use X.509 credential technology to authenticate the user. This type of credential, however, does not offer the same privacy preserving features as anonymous credential systems.

This section presents a strategy that tackles these issues using the setup presented in the previous chapter. The PIN of the user is entered via the TEE application that runs on the workstation of the user. This prevents malware on the workstation from intercepting, and consequently abusing the PIN. To improve the user's privacy, the TEE application also acts as an authentication proxy. The eID authenticates towards the TEE application that subsequently attests the authentication towards the remote service provider. The attestation is realized using privacy preserving credential technologies. It, hence, does not require releasing uniquely identifiable information, allowing multi-show unlinkability. The TEE application can choose to release only a subset of the information contained in the eID, realizing selective disclosure.

The *contribution* of this section is twofold. First, it presents a solution that allows users to securely enter their PIN via a workstation to activate the authentication credentials on their smart card. The solution further extends existing smart card assisted authentication technology based on X.509 credentials with privacy-preserving features such as multi-show unlinkability and selective disclosure. It can, for instance, be used to increase the privacy

and security of several governmental electronic identity infrastructures [22, 110, 70, 75]. It, therefore, applies the secure virtualization technologies contained on the workstation. Second, a prototype implementation of this system is presented to validate our solution. For the prototype, a CCID stack is added to an existing framework for building SEE applications. The CCID standard specifies the communication protocols between an USB host and a smart card reader. It is supported by most commercial smart card readers.

8.2.1 Design

This section first lists the different actors in the system, followed by the requirements and a general description of the system. Finally, a detailed description of the protocols is presented.

Roles

We assume a user (U) working on a workstation (W) that runs a legacy operating system and supports TEE technologies for running a trusted application (PAL). A smart card reader is attached to the workstation. The user also carries a smart card (SC), from a rolled-out eID infrastructure, that contains an X.509 authentication credential (i.e. X.509 certificate and private key) of the user. This is used to authenticate the user to a remote service provider (SP) via the workstation. Before authentication, the user needs to unlock the credential on his smart card using his PIN.

Requirements

S_1 Malware running on the workstation cannot intercept the PIN of the user.

S_2 Malicious software cannot mislead the user into approving a malicious signing transaction/authentication attempt or disclosing more information than desired by the user.

S_3 The user can select which attributes are disclosed to the service provider. If no uniquely identifying attributes are disclosed, service providers cannot link different authenticated sessions.

S_4 The service provider is assured that the attributes released during authentication are certified by a trusted CA.

General Approach

The user authenticates towards a remote service provider via a workstation using his smart card. Instead of using the X.509 credentials on the smart card to authenticate towards the service provider, they are used to authenticate towards a trusted *PAL* running on the workstation. To this end, the public key of the CA (pk_{ca}) is embedded in the *PAL* binary. During authentication, the user is required to enter his PIN. Instead of entering the PIN via the regular operating system, the OS is suspended and the trusted *PAL* is started to handle the input of the PIN and the authentication by the smart card. This protects the PIN from being intercepted by malware running on the workstation. The *PAL* verifies the authentication and can, subsequently, selectively attest the attributes, obtained during the authentication, towards the service provider using the *quote* functionality of the TPM. Since the TPM's anonymous attestation capabilities (i.e. DAA) can be used for the attestation, no information but the selectively disclosed attributes, the fact that the user had a valid credential and that the filtering was performed by a known trusted application is released. The trusted *PAL* not only acts as an authentication proxy between the eID and the service provider, it also gives the user control over the attributes released to the specified service provider. This prevents malware running on the workstation from misleading the user into approving malicious transactions.

Adversary and Trust Model

The same adversary model as the previous chapter is used. The *PAL* is trusted by both the user and the service provider. The user trusts the *PAL* for informed consent regarding the pending authentication. The service provider trusts the *PAL* for the provisioning of the user's information.. This trust is supported by the attestation of the *PAL* towards the service provider and mobile. The state of the trusted *PAL* can be certified by a trusted third party. The *PAL* should also be open source so that independent developers can verify that the certified state correctly represents the desired functionality.

Protocols

Prerequisites. The user has gone through the enrollment phase as described in the previous chapter. After the enrollment phase, the user selected an authentication picture (img_u) on the mobile device which was subsequently encrypted with the public of the *PAL* (enc_{img}) and transferred to the workstation where it was stored. Hence, the user knows the trusted *PAL* is

running if the correct picture (img_u) is shown. The service provider obtained the certified PCR state of the trusted *PAL*.

User Authentication. Figure 8.2 presents the protocol for authenticating the user towards a remote service provider. First, the user requests access to a protected resource from the remote service provider (1). The provider responds with an authentication request containing its certificate, the attribute request (i.e. a list of identifiers representing the attributes that need to be released) and an attestation challenge (2). Subsequently, the *PAL* is started and the attribute request is passed as a parameter, together with the encrypted authentication image (enc_{img}) and the certificate of the service provider ($cert_{sp}$) (3). The *PAL* now unseals enc_{img} (4). The user is subsequently informed by the *PAL* regarding the pending authentication and requested to enter his PIN. Meanwhile, the user's unique image is shown to indicate to the user that the trusted environment is running (5). The user can, hence, trust the information shown on the display. To acknowledge the authentication, the user enters his PIN (6). The *PAL* can now unlock the credentials on the smart card using the obtained PIN and send an authentication challenge to the smart card (7). If the PIN verification succeeds (8), the card signs the authentication challenge and transfers the resulting signature (sig) and the authentication certificate ($cert_u$) to the *PAL* (9-10). The *PAL* now verifies the authentication (11-12). If the authentication verification succeeds, the *PAL* extracts the requested attributes ($atts$) from the authentication certificate (13). The *PAL extends* its state with the requested attributes and the certificate of the service provider (14). The *PAL* ends its execution and returns the user's attributes back to the regular OS, that resumes its operation (15). The OS now performs a *quote* operation on the state resulting from the *PAL* execution (16). This quote attests towards the service provider that the trusted *PAL* indeed obtained the user's attributes from a valid smart card (i.e. the *PAL* state is extended with the user's attributes) and that the user was shown the correct information about the service provider (i.e. the *PAL* state is extended with the service provider's certificate). The resulting quote is sent to the service provider along with $atts$ (17). The service provider now verifies the quote and, hereby, checks the authenticity of the received attributes (18). Upon success, the service provider grants access to the requested resource (19).

8.2.2 Realization and Validation

This section validates and illustrates the feasibility of our approach by developing a prototype for the Belgian eID infrastructure. Several service providers, both commercial and governmental, use the eID infrastructure to

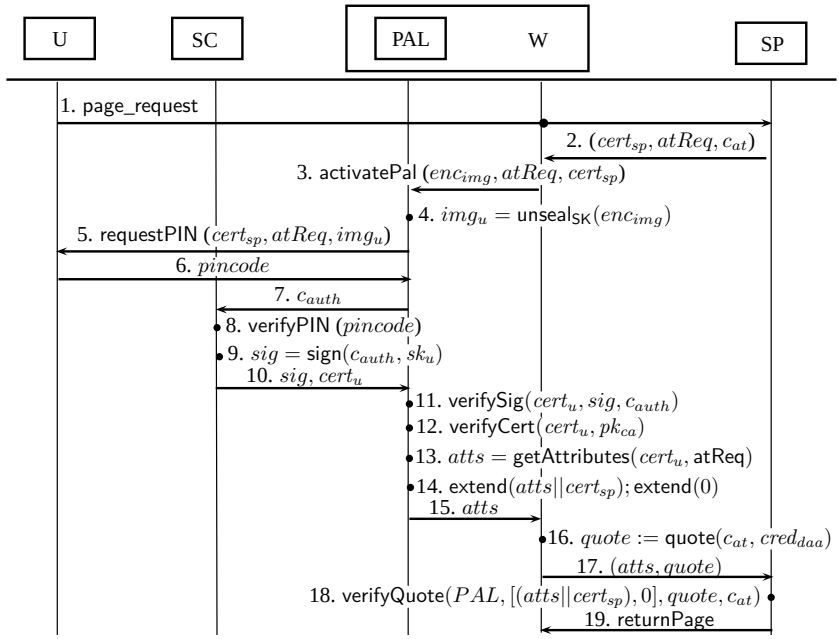

Figure 8.2: Privacy friendly authentication using X.509 credentials (U: User, SE: Secure Element, PAL: Piece of Application Logic, W: Workstation, SP: Service Provider).

handle user authentication. To test the compatibility with these existing service providers, the prototype is forced to release the entire authentication certificate. Adding the selective disclosure parts of the protocol, however, does not pose any additional technical challenges but requires some additional logic at the service provider.

The Belgian electronic identity card contains an RSA private key and an X.509 certificate that is used for user authentication. This credential is activated via the PIN of the user. The government provides a middleware package that needs to be installed on the workstation of the user. It offers an interface (i.e. `PKCS#11`) that other applications (e.g. the browser) can use to support eID authentication. It handles the communication to the card and allows the user to enter his PIN. For authentication towards a remote service provider via the browser, the card is typically used to set up a mutually authenticated HTTPS session between the browser and the service provider. The browser uses the `PKCS#11` interface made available by the middleware during the HTTPS handshake.

The same hardware is used as in the previous chapter for the realization of the different entities. A smartphone (*MD*) is used in the enrollment procedure during which the user seals an authentication image to the state of the trusted *PAL*. On the workstation, Ubuntu 12.04 is running. The `PKCS#11` module of the existing middleware has been modified so that the operations requiring a verified PIN are delegated to the trusted *PAL*. The software architecture of the prototype software on the workstation is shown in Figure 8.3. When Firefox is used to access a remote protected resource, the browser calls the `PKCS#11` token to authenticate the user. The `PKCS#11` token calls the `SEE driver` to delegate any operations requiring a verified PIN to the *PAL*. For the *PAL* application, a CCID driver was implemented on top of the USB-UHCI stack. This allows the *PAL* to communicate with the eID inserted in the smart card reader. The *PAL* further contains a minimal version of the eID middleware to unlock and use the credentials on the eID.

Our *PAL* application consists of three main components. The CCID driver and eID middleware (646 lines of code) run on top of the USB-UHCI stack (1214 lines of code). The implementation of the *PAL* protocol itself consists of 781 lines of code. This adds a total of 2641 lines of code to the Flicker framework, preserving a limited trusted computing base. Although adding the selective disclosure part of the protocol (filtering of attributes, parsing of the access control policy, etc) will slightly increase the TCB, it will remain sufficiently small to suggest that it can be formally verified not to contain common vulnerabilities such as buffer overflows. In terms of performance, the whole authentication process takes about seven seconds compared to about two seconds for the regular authentication process.

The prototype has been tested with several existing service providers. The system only requires the user to install the modified middleware, no modification of the browser or service provider are required. The selective disclosure functionality, however, does require adding some additional logic to the service provider. A similar implementation approach can be taken for other European eID cards as many (e.g. Italy [22], Spain [110] and Portugal [70]) use a similar design to that of the Belgian eID.

8.2.3 Evaluation

Requirements Review

During authentication the user is assured that the trusted application is running by displaying the personal image. Since the personal image is only accessible to the trusted *PAL* and the user only enters his PIN if the correct image is

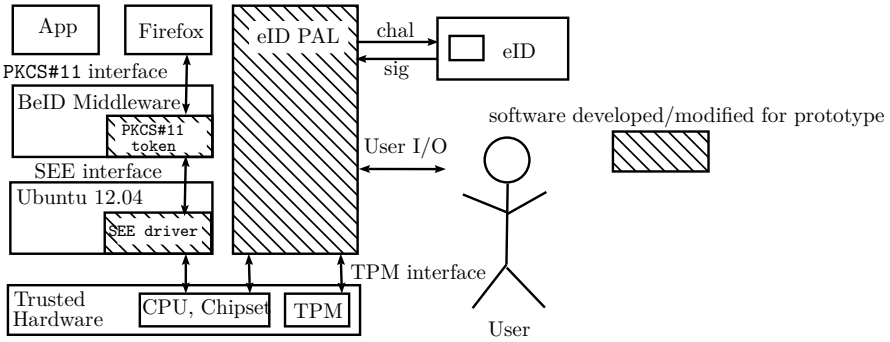

Figure 8.3: Overview of the software architecture of the prototype.

shown, the *PAL* is in full control of the workstation during the entering of the PIN. Hence, malware running in the OS cannot intercept the entered PIN. This satisfies security requirement S_1. Similarly, the user is assured that the provided information about the pending authentication is correct. The *PAL* binds the authentication proof to the service provider shown to the user by extending its state with the service provider's certificate. This satisfies security requirement S_2.

During the authentication of the eID to the *PAL*, the latter obtains all of the user's attributes contained on the card. The *PAL*, however, filters the received information and only releases the attributes required to satisfy the access control policy of the service provider. This allows the user to selectively disclose attributes. The DAA protocol used for attestation does not require releasing uniquely identifying information, satisfying requirement S_3.

The *PAL* verifies the authenticity of the used smart card via a challenge-response protocol using the key of the CA which is embedded in the *PAL* binary. The *PAL* only continues the authentication process if a valid smart card is used. The service provider is assured that a trusted *PAL* checked the validity of the used smart card via the attestation protocol. Hence, the service provider is assured that only users with a valid smart card can authenticate. If attributes are released, they are included in the assertion. The service provider, hence, is assured that the released attributes are obtained from a genuine smart card. This satisfies requirement S_4.

Security and Privacy Considerations

Currently, the user trusts his mobile device to correctly verify the integrity of the *PAL*. This additional hardware component allows the system to be used with existing rolled out identity card infrastructures. It however, also introduces an additional trust relationship (i.e. user and mobile device). If the software on the smart card would be extended to include the verification of the *PAL* environment, the user would no longer need a mobile device to use the system. Moreover, the software running on the smart card is more trustworthy than software running on a mobile device.

Identity cards that have been stolen or otherwise compromised can be revoked. This has no impact on the *trusted PIN input* system as the service provider receives the authentication certificate and can, hence, check the revocation status. In the *privacy friendly authentication* system, however, the service provider only receives the attributes released by the *PAL*. The *PAL*, therefore, is responsible for checking the revocation status of the card. If the revocation list contains a limited number of serial numbers, it can be passed along to the *PAL* as an argument. The *PAL* checks the signature on the revocation list and then uses the revocation list to check the revocation status of the eID. In case the revocation list is too large, the *PAL* generates a nonce that is transferred to an OCSP server together with the serial number of the eID. The nonce should be transferred via the regular OS to the OCSP server to ensure that the *PAL* doesn't have to contain networking drivers which would bloat the TCB. The response is subsequently transferred back to the *PAL* that can now verify the revocation status of the card. The use of OCSP, however, also has privacy implications. Timing attacks could be used by the relying party and OCSP server to link the identity of the user to the service request.

8.3 Conclusion

This chapter presented two case studies in which the system presented in the previous chapter is applied to increase the security and/or privacy of identity management systems.

In the first case study a new solution for activating credentials bound to its owner by means of biometrics. It assures that users are physically present when their credentials are used, effectively impeding credential sharing and abuse by theft. Moreover, credential abuse by malware is prevented by isolating the credential operations in a secure environment on a workstation. Apart from the hardware support available in modern commodity workstations, no

additional infrastructure is required. The system can be applied to general client-server authentication use-cases or dedicated use-cases such as electronic identity systems or eBanking. A prototype implementation demonstrates the feasibility of our system. In future research, this approach can be extended to include the verification of contextual information such as geographical data. A service provider could require the workstation to be located in a certain country, providing location based/restricted services.

The second case study presented a system that can be applied to increase both the security and privacy of existing smart card based authentication systems that use X.509 credentials. The security enhancement allows users to securely enter their PIN via the workstation to activate the authentication credentials on the smart card. The PIN is entered via an application running in a trusted execution environment on the workstation. The system further allows the user to select which attributes contained in the authentication credential to disclose towards the service provider, increasing the privacy of the user. The attributes are attested by the application running in the trusted execution environment, hereby assuring the service provider that the released information is correct. A prototype using the Belgian eID card as authentication token is presented, illustrating the feasibility of the system.

Chapter 9

General Conclusion

As stated in the introduction, this thesis tries to contribute to answering the question *how hardware based security technologies, more specifically secure element and trusted execution environment technologies, can be applied to realize the diverse set of requirements inherent to identity management systems and technologies.* This thesis consists of three parts that each use hardware security technologies to tackle a specific set of requirements. This chapter reflects on the use of those hardware security technologies.

9.1 Hardware-Backed Authentication using Symmetric Key Technology

In the first part, a security architecture was presented that allows resource constrained devices to establish an authenticated channel with powerful devices. Data can be securely exchanged over this channel. The constrained device can enforce fine-grained access control policies based on the information obtained during authentication. The system specifically aims at settings in which low-cost constrained devices are used that can only execute symmetric key operations and have to securely exchange data with general purpose devices (e.g. smartphones or laptops) from different users, each with a specific set of privileges.

In open environments, an online authority has to issue credentials to powerful devices on-the-fly. In application domains with a limited scope, key management can be considerably simplified by storing a shared secret key in

a tamper-resistant module on the constrained device. This avoids the need for an online key authority. Several types of tamper-resistant modules can be used (e.g. secure element, apply a secure casing, using SoC memory). The key never leaves the protected hardware, ensuring that it is not possible for an attacker to retrieve or modify it without resorting to invasive or side-channel attacks. The limited scope of the closed environment and the low-value application domain should ensure that the cost for extracting the key is higher than the potential gains for an attacker. Depending on the type of tamper-resistant module used, a dedicated mechanism can be used to update the shared key and, consequently, mitigate the impact of compromised keys.

9.2 SE-Backed Identity Management

9.2.1 Summary

Most identity management systems rely on digital signature algorithms to assert the validity of attributes to service providers. The identity management system proposed in part two relies on a trusted application running on a tamperproof secure element. The service provider is assured that the received information originates from a genuine secure element in the system by establishing an authenticated secure channel with the secure element. The tamperproofness ensures that an attacker cannot directly access the memory of the secure element to extract or modify information. This assures the service provider that the information received over the secure channel from the trusted secure element application is genuine.

The use of a secure element has several disadvantages. First, secure elements are typically resource constrained devices. The limited computational power and storage and the low bandwidth communication have a significant impact on the performance of the system. Several measures can be taken to optimize performance. The system should be designed to make the secure element application as efficient as possible. For instance, the proof-of-concept implementation uses an elliptic curve cryptography-based key agreement protocol and the RSA signature verification algorithm to establish a secure authenticated session with providers. The implementation further uses efficient data structures and encoding rules and limits the number of asymmetric key operations on the secure element (e.g. avoid the verification of intermediate CA certificates, use a caching system to avoid having to aggregate many attributes from identity providers). The secure element can also precalculate values (e.g. ephemeral keys), periodically execute the (re)validation protocol and prefetch commonly used attributes from identity providers during idle

periods to increase the performance during authentication. Allowing sufficient personalization can also have a beneficial impact on the performance. For instance, allowing users to trigger the card to prefetch specific attributes can limit the number of contacted identity providers during authentication. To support large attribute values and keep the communication overhead to the secure element limited, large attribute values could be transferred via the middleware while a cryptographic hash is asserted by the secure element.

Second, secure elements typically rely on additional hardware to interface with the user and providers. To establish a trusted path between the user and the secure element, dedicated certified hardware could be used. An alternative approach is to implement (a part of) the middleware in a trusted execution environment. The advantage of this approach is that it does not require separate hardware since TEE technologies are (starting to be) integrated in commodity devices such as workstation, laptops and smartphones. The third part of this text investigates how TEE technologies on workstation and laptops can be used to provide a secure trustworthy interface to the secure element.

While most applications rely on a secure element to provide additional security, the identity management system presented in this part relies on the tamperproofness of the secure element for its core security requirements. Whether or not this assumption holds in practice in the context of an identity management system remains an open question. Especially since the build-in privacy-preserving features impede the discovery and revocation of hacked secure elements. A system with similar trust assumptions that has recently been deployed is the German electronic identity infrastructure [32]. This system can provide a good indication whether or not systems that rely on a secure element for more than just an additional layer of security are feasible in practice.

9.2.2 Future work

The current implementation only provides a proof of concept of the presented system. The proof of concept only supports a limited number of attributes, providers and policy options. Hence, to assess how the system would behave in practice, the proof of concept should be extended to evaluate the impact of increasing the number of providers and policy options on the performance of the system. Currently, the proof of concept uses the trust model with a single audit authority. However, as discussed in Chapter 5, this model is less suited to accurately represent the trust relationships between providers when the number of providers increases. The introduction of multiple audit authorities increases the complexity and will, hence, affect performance. After extending the proof of concept, the usability of the system can be evaluated via user studies.

Although secure elements provide protection against hardware attacks, given sufficient investment hardware attacks are feasible. Hence, to prevent a *Break Once, Run Everywhere* attack, the same common key is only shared by a batch of cards. The number of secure elements on which the same key is used is crucial factor in the system. The number of secure elements in a batch should be chosen so that the impact of a hacked secure element is sufficiently low. The impact is largely determined by the cost for hardware (i.e. new cards need to be issued), logistics (i.e. the process of issuing a new batch of cards), user convenience (i.e. all the users with a card in the affected batch need to replace their card) and provider integrity (i.e. the hacked key could be used to gain access to services). However, lowering the number of cards in a batch negatively impacts the privacy of the user since it makes it easier to uniquely identify the user. Hence, for determining the number of cards in a batch, the system administrator will need to balance these two factors. A thorough comparison of the cost, scalability, usability and privacy of a similar system based on anonymous credentials provides an interesting topic for future research.

Since the system relies on both the hardware and software integrity of the secure element, also software attacks have to be considered. Formal verification of the code running on the secure element would, hence, provide a more solid foundation for deploying the system.

9.3 TEE-Backed Identity Management

9.3.1 Summary

In this part, the trusted execution environment that can be established on commodity workstations and laptops using TPM-based technologies is used to increase the security and privacy of existing identity management technologies. Two complementary case studies are presented.

The first case study focuses on the prevention of credential abuse through sharing or theft. The trusted execution environment is used to assert to the service provider that the owner of the credential approved the transaction (i.e. informed consent). The user's credential is, therefore, bound to one or more of his biometric traits. The TEE application informs the user about the pending transaction and verifies the binding between the used credential and the user using a live reading of the biometric trait(s) of the user.

The second case study focuses on increasing the security and privacy of authentication infrastructures that rely on a smart card containing an X.509 credential and the identity information of the user. Several existing

governmental electronic identity infrastructures use such an approach. The system increases the security by having the user enter his passcode via a TEE application. This prevents malware running on the regular operating system from intercepting the passcode. Several governmental identity systems only implement limited privacy-preserving features. The TEE application gives the user more control over the disclosure of his information. It offers features such as multi-show unlinkability and selective disclosure. The electronic identity card, therefore, authenticates to the TEE application that, subsequently, filters the obtained information and attests a user-approved subset to the service provider.

For both case studies a proof-of-concept is implemented to illustrate the feasibility of the presented system. The current implementation intensively uses the TPM for the realization of the trusted execution environment. The resource constrained nature of the TPM, hence, has a significant impact on the performance of the system. A promising approach for significantly improving the performance of TEE application is the use of a virtual TPM. During boot, a small measured hypervisor is loaded that contains a virtual TPM. This virtual TPM runs on the general purpose processor and is, hence, much less constrained than the hardware TPM. The hardware TPM is used during boot to ensure the integrity of the hypervisor and software TPM. The system, hence, provides better performance with a small increase in the trusted computing base (i.e. the hypervisor and virtual TPM).

Currently, there is no central authority that issues TPM certificates. Most TPMs are shipped uninitialized and uncertified. Hence, before a user can start using the TPM embedded in his device it must be initialized. During the trusted enrollment procedure, the user can establish trust in the TPM's credential generated during initialization. Hence, to establish a secure interface to the user, the certification of the TPM is a less stringent requirement. However, if the TPM is used to assert data towards a remote party, the remote party needs to be assured that a valid TPM was used. The most scalable application domain for use cases using the assert functionality of the TPM towards remote parties are those in which a central authority is responsible for managing the hardware (e.g. the corporate application domain).

9.3.2 Future Work

Currently, the system uses the Flicker framework to realize the trusted execution environment of the workstation. More recent work [190, 179, 180], further developed that approach and addressed the performance. The systems presented in Chapter 8 could be implemented using these new systems to

improve performance. Apart from the performance, also the switching between the two environments affects the usability. A user study would be able to assess whether or not the system meets the usability requirements of users and what aspects should be further improved. Apart from increasing the performance, also the quality of the software would need to be improved and tested before the system could be used in practice. Formal methods [117, 94, 133] could be used to ensure that the application running in the trusted environment is free from several commonly exploited types of bugs.

Currently the mobile device is used to measure the state of the system. Mobile devices are computational rich devices that run a rich software stack and third-party software, making them vulnerable to abuse via exploits. The mobile device could be replaced with a dedicated USB device with the sole purpose of measuring the software state of the workstation. An alternative would be to realize a trusted environment on the mobile device. If a biometric scanner is embedded in the mobile device, which is currently the case in some high-end smartphones, the trusted application could be migrated from the workstation to the trusted environment on the mobile device. Since the biometric scanner is actually embedded in the casing of the mobile device and not a plug-and-play device as is often the case with workstations and laptops, the system can provide more assurance that the biometric scan is an actual live reading of the user holding the device.

Bibliography

[1] The Apache Tomcat project. http://tomcat.apache.org. pages 44, 89, 116

[2] Eikon II UPEK fingerprint reader. http://www.digitalpersona.com/Fingerprint-Biometrics/Fingerprint-Readers/Fingerprint-Readers. pages 115

[3] Ghostery. https://www.ghostery.com/en. pages 22

[4] The Higgins project homepage. https://www.eclipse.org/higgins. pages 93

[5] Java Cards. http://www.oracle.com/technetwork/java/javacard/specs-138637.html. pages 44, 78

[6] Lightbeam. https://www.mozilla.org/en-US/lightbeam. pages 22

[7] Memcached. http://memcached.org. pages 89

[8] Mobile security card SE 1.0. http://www.gi-de.com. pages 56, 76, 78, 85, 89

[9] Multi-format 1D/2D barcode image processing library with clients for Android, Java. https://github.com/zxing/zxing. pages 89

[10] Noscript. http://noscript.net. pages 22

[11] Polarssl. https://polarssl.org. pages 102, 116

[12] Secure element evaluation kit for the Android platform. http://code.google.com/p/seek-for-android. pages 89

[13] The spring security project. http://static.springsource.org/spring-security/site. pages 89, 116

[14] The TrouSerS TCG software stack. `http://sourceforge.net/projects/trousers`. pages 116

[15] Abdul-Rahman, A., and Hailes, S. A distributed trust model. In *Proceedings of the 1997 Workshop on New Security Paradigms* (New York, NY, USA, 1997), NSPW '97, ACM, pp. 48–60. pages 11

[16] Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., and Preneel, B. FPDetective: Dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (New York, NY, USA, 2013), CCS '13, ACM, pp. 1129–1140. pages 23

[17] Adida, B. Beamauth: Two-factor web authentication with a bookmark. In *Proceedings of the 14th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2007), CCS '07, ACM, pp. 48–57. pages 10

[18] Ahn, G.-J., and Ko, M. User-centric privacy management for federated identity management. In *COLCOM '07: Proceedings of the 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 187–195. pages 2, 14

[19] Aloul, F., Zahidi, S., and El-Hajj, W. Two factor authentication using mobile phones. In *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on* (2009), pp. 641–644. pages 10

[20] Ameen, M., Liu, J., and Kwak, K. Security and privacy issues in wireless sensor networks for healthcare applications. *J. Med. Syst. 36*, 1 (Feb. 2012), 93–101. pages 36

[21] ANSI. Public key cryptography for the financial services industry: Agreement of symmetric keys using discrete logarithm cryptography. `www.ansi.org`. pages 77, 81

[22] Arcieri, F., Ciclosi, M., Fioravanti, F., Nardelli, E., and Talamo, M. The Italian electronic identity card: A short introduction. In *Proceedings of the 2004 Annual National Conference on Digital Government Research* (2004), dg.o '04, Digital Government Society of North America, pp. 73:1–73:2. pages 15, 123, 127

[23] ARM. Trustzone. `http://www.arm.com/products/processors/technologies/trustzone/index.php`. pages 19

[24] Backes, M., Gerling, S., Hammer, C., Maffei, M., and von Styp-Rekowsky, P. AppGuard: Enforcing user requirements on Android apps. In *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (Berlin, Heidelberg, 2013), TACAS'13, Springer-Verlag, pp. 543–548. pages 23

[25] Baldimtsi, F., and Lysyanskaya, A. Anonymous credentials light. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (New York, NY, USA, 2013), CCS '13, ACM, pp. 1087–1098. pages 13

[26] Balfanz, D., and Felten, E. W. Hand-held computers can be better smart cards. In *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8* (Berkeley, CA, USA, 1999), SSYM'99, USENIX Association, pp. 2–2. pages 21, 122

[27] Bangerter, E., Camenisch, J., and Lysyanskaya, A. A cryptographic framework for the controlled release of certified data. In *Proceedings of the 12th International Conference on Security Protocols* (Berlin, Heidelberg, 2006), SP'04, Springer-Verlag, pp. 20–42. pages 17

[28] Batina, L., Lee, Y. K., Seys, S., Singelée, D., and Verbauwhede, I. Extending ECC-based RFID authentication protocols to privacy-preserving multi-party grouping proofs. *Personal Ubiquitous Comput. 16*, 3 (Mar. 2012), 323–335. pages 27, 28

[29] Bauer, D., Blough, D. M., and Cash, D. Minimal information disclosure with efficiently verifiable credentials. In *Proceedings of the 4th ACM Workshop on Digital Identity Management* (New York, NY, USA, 2008), DIM '08, ACM, pp. 15–24. pages 13

[30] Bay, A., Boureanu, I., Mitrokotsa, A., Spulber, I., and Vaudenay, S. The Bussard-Bagga and other distance-bounding protocols under attacks. In *Information Security and Cryptology*, M. Kutylowski and M. Yung, Eds., vol. 7763 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 371–391. pages 33

[31] Bellare, M., and Namprempre, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology 21* (2008), 469–491. pages 9, 77

[32] Bender, J., Kügler, D., Margraf, M., and Naumann, I. Privacy-friendly revocation management without unique chip identifiers for the german national id card. *Computer Fraud & Security 9* (Sept. 2010), 14–17. pages 2, 15, 56, 133

[33] Beresford, A. R., Rice, A., Skehin, N., and Sohan, R. Mockdroid: Trading privacy for application functionality on smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications* (New York, NY, USA, 2011), HotMobile '11, ACM, pp. 49–54. pages 23

[34] Bichsel, P., Camenisch, J., De Decker, B., Lapon, J., Naessens, V., and Sommer, D. Data-minimizing authentication goes mobile. In *Proceedings of the 13th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security* (Berlin, Heidelberg, 2012), CMS'12, Springer-Verlag, pp. 55–71. pages 13, 17, 71, 111

[35] Bichsel, P., Camenisch, J., Gross, T., and Shoup, V. Anonymous credentials on a standard Java Card. In *Proceedings of the 16th ACM conference on Computer and communications security* (New York, NY, USA, 2009), CCS '09, ACM, pp. 600–610. pages 2, 13, 111, 120

[36] Biddle, R., Chiasson, S., and Van Oorschot, P. Graphical passwords: Learning from the first twelve years. *ACM Comput. Surv. 44*, 4 (Sept. 2012), 19:1–19:41. pages 10

[37] Blanton, M., and Hudelson, W. M. P. Biometric-based non-transferable anonymous credentials. In *Proceedings of the 11th international conference on Information and Communications Security* (Berlin, Heidelberg, 2009), ICICS'09, Springer-Verlag, pp. 165–180. pages 111, 120

[38] Blaze, M., Feigenbaum, J., and Lacy, J. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 1996), SP '96, IEEE Computer Society, pp. 164–. pages 11

[39] Bleumer, G. Biometric yet privacy protecting person authentication. In *Information Hiding*, vol. 1525 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1998, pp. 99–110. pages 111, 120

[40] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., and Vikkelsoe, C. Present: An ultra-lightweight block cipher. In *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems* (Berlin, Heidelberg, 2007), CHES '07, Springer-Verlag, pp. 450–466. pages 27, 33

[41] Boukayoua, F., Vossaert, J., Decker, B., and Naessens, V. Claim-based versus network-based identity management: A

hybrid approach. In *Security and Privacy in Mobile Information and Communication Systems*, A. U. Schmidt, G. Russello, I. Krontiris, and S. Lian, Eds., vol. 107 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer Berlin Heidelberg, 2012, pp. 38–50. pages 5, 87

[42] Boukayoua, F., Vossaert, J., Decker, B., and Naessens, V. Using a smartphone to access personalized Web services on a workstation. In *Privacy and Identity Management for Life*, J. Camenisch, B. Crispo, S. Fischer-Hübner, R. Leenes, and G. Russello, Eds., vol. 375 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2012, pp. 144–156. pages 5, 87

[43] Boulos, M. K., Rocha, A., Martins, A., Vicente, M., Bolz, A., Feld, R., Tchoudovski, I., Braecklein, M., Nelson, J., Ó Laighin, G., Sdogati, C., Cesaroni, F., Antomarini, M., Jobes, A., and Kinirons, M. CAALYX: a new generation of location-based services in healthcare. *International Journal of Health Geographics 6*, 1 (2007), 1–6. pages 36

[44] Boureanu, I., Mitrokotsa, A., and Vaudenay, S. Secure and lightweight distance-bounding. In *Lightweight Cryptography for Security and Privacy*, G. Avoine and O. Kara, Eds., vol. 8162 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 97–113. pages 33

[45] Boyd, C., and Mathuria, A. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, 2003. pages 25

[46] Brands, S., Demuynck, L., and De Decker, B. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *Proceedings of the 12th Australasian Conference on Information Security and Privacy* (Berlin, Heidelberg, 2007), ACISP'07, Springer-Verlag, pp. 400–415. pages 5, 12

[47] Brands, S. A. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000. pages 1, 12, 13

[48] Brasser, F. F., Bugiel, S., Filyanov, A., Sadeghi, A.-R., and Schulz, S. Softer smartcards - usable cryptographic tokens with secure execution. In *Financial Cryptography* (2012), A. D. Keromytis, Ed., vol. 7397 of *Lecture Notes in Computer Science*, Springer, pp. 329–343. pages 74, 95, 98, 99

[49] Brickell, E., Camenisch, J., and Chen, L. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security* (New York, NY, USA, 2004), CCS '04, ACM, pp. 132–145. pages 97, 111

[50] Butun, I., and Sankar, R. A brief survey of access control in wireless sensor networks. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE* (Jan 2011), pp. 1118–1119. pages 28

[51] Camenisch, J. Protecting (anonymous) credentials with the trusted computing group's tpm v1.2. In *Security and Privacy in Dynamic Environments*, S. Fischer-Hübner, K. Rannenberg, L. Yngström, and S. Lindskog, Eds., vol. 201 of *IFIP International Federation for Information Processing*. Springer US, 2006, pp. 135–147. pages 111, 120

[52] Camenisch, J., Dubovitskaya, M., Lehmann, A., Neven, G., Paquin, C., and Preiss, F.-S. Concepts and languages for privacy-preserving attribute-based authentication. In *Policies and Research in Identity Management*, S. Fischer-Hübner, E. Leeuw, and C. Mitchell, Eds., vol. 396 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2013, pp. 34–52. pages 85

[53] Camenisch, J., and Lysyanskaya, A. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology* (London, UK, UK, 2001), EUROCRYPT '01, Springer-Verlag, pp. 93–118. pages 1, 12, 13

[54] Camenisch, J., and Lysyanskaya, A. A signature scheme with efficient protocols. In *Proceedings of the 3rd International Conference on Security in Communication Networks* (Berlin, Heidelberg, 2003), SCN'02, Springer-Verlag, pp. 268–289. pages 12, 13

[55] Camenisch, J., Mödersheim, S., Neven, G., Preiss, F.-S., and Sommer, D. A card requirements language enabling privacy-preserving access control. In *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies* (New York, NY, USA, 2010), SACMAT '10, ACM, pp. 119–128. pages 18, 85

[56] Camenisch, J., shelat, a., Sommer, D., Fischer-Hübner, S., Hansen, M., Krasemann, H., Lacoste, G., Leenes, R., and Tseng, J. Privacy and identity management for everyone. In *Proceedings of the 2005 Workshop on Digital Identity Management* (New York, NY, USA, 2005), DIM '05, ACM, pp. 20–27. pages 17

[57] Camenisch, J., and Shoup, V. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology - CRYPTO 2003*, D. Boneh, Ed., vol. 2729 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 126–144. pages 13

[58] Camenisch, J., and Van Herreweghen, E. Design and implementation of the Idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2002), CCS '02, ACM, pp. 21–30. pages 1, 13, 110, 120

[59] Castle, B. Legion of the Bouncy Castle: The Bouncy Castle crypto APIs. `http://www.bouncycastle.org`. pages 27, 78

[60] Chadwick, D. Federated identity management. In *Foundations of Security Analysis and Design V*, A. Aldini, G. Barthe, and R. Gorrieri, Eds., vol. 5705 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2009, pp. 96–120. pages 2, 14, 52

[61] Chadwick, D. W., Inman, G., and Klingenstein, N. A conceptual model for attribute aggregation. *Future Gener. Comput. Syst. 26*, 7 (July 2010), 1043–1052. pages 14

[62] Chadwick, D. W., Inman, G., and Klingenstein, N. A conceptual model for attribute aggregation. *Future Gener. Comput. Syst. 26*, 7 (July 2010), 1043–1052. pages 51, 64

[63] Chan, H., and Perrig, A. Pike: peer intermediaries for key establishment in sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE* (March 2005), vol. 1, pp. 524–535 vol. 1. pages 25, 27

[64] Chaum, D. Blind signatures for untraceable payments. In *Advances in Cryptology*, D. Chaum, R. Rivest, and A. Sherman, Eds. Springer US, 1983, pp. 199–203. pages 12

[65] Chaum, D. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM 28*, 10 (Oct. 1985), 1030–1044. pages 12

[66] Chaum, D., and Pedersen, T. P. Wallet databases with observers. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 1993), CRYPTO '92, Springer-Verlag, pp. 89–105. pages 111, 120

[67] Chiasson, S., Oorschot, P., and Biddle, R. Graphical password authentication using cued click points. In *Computer Security - ESORICS*

2007, J. Biskup and J. López, Eds., vol. 4734 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 359–374. pages 10

[68] Corporation, I. LaGrande technology preliminary architecture specification. Intel Publication no. D52212, May 2006. pages 2, 20, 97

[69] Cremers, C., Rasmussen, K. B., Schmidt, B., and Capkun, S. Distance hijacking attacks on distance bounding protocols. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2012), SP '12, IEEE Computer Society, pp. 113–127. pages 33

[70] Crocker, P., Nicolau, V., and mao Melo de Sousa, S. Sniffing with Portuguese identity card for fun and profit. In *Proc European Conference on Information Warfare and Security - ECIW*, vol. 9. Academic Conferences Limited, 2010, pp. 43–56. pages 15, 123, 127

[71] Cryptopp. Speed comparison of popular crypto algorithms. `http://www.cryptopp.com/benchmarks.html`. pages 77

[72] Czeskis, A., Koscher, K., Smith, J. R., and Kohno, T. RFIDs and secret handshakes: Defending against ghost-and-leech attacks and unauthorized reads with context-aware communications. In *Proceedings of the 15th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2008), CCS '08, ACM, pp. 479–490. pages 33

[73] Damgård, I. Efficient concurrent zero-knowledge in the auxiliary string model. In *Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques* (Berlin, Heidelberg, 2000), EUROCRYPT'00, Springer-Verlag, pp. 418–430. pages 13

[74] De Cannière, C. Trivium: A stream cipher construction inspired by block cipher design principles. In *Information Security*, S. Katsikas, J. López, M. Backes, S. Gritzalis, and B. Preneel, Eds., vol. 4176 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 171–186. pages 27

[75] De Cock, D., Wolf, C., and Preneel, B. The Belgian electronic identity card (overview). In *Sicherheit*, J. Dittmann, Ed., vol. 77 of *LNI*. GI, 2006, pp. 298–301. pages 15, 123

[76] De Cock, D., Wouters, K., and Preneel, B. Introduction to the Belgian eID card. In *Public Key Infrastructure*, S. Katsikas, S. Gritzalis, and J. López, Eds., vol. 3093 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 1–13. pages 15

[77] Denning, T., Borning, A., Friedman, B., Gill, B. T., Kohno, T., and Maisel, W. H. Patients, pacemakers, and implantable defibrillators: Human values and security for wireless implantable medical devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2010), CHI '10, ACM, pp. 917–926. pages 36

[78] Deswarte, Y., and Gambs, S. A proposal for a privacy-preserving national identity card. *Trans. Data Privacy 3*, 3 (Dec. 2010), 253–276. pages 111, 120

[79] Deswarte, Y., and Gambs, S. The challenges raised by the privacy-preserving identity card. In *Cryptography and Security: From Theory to Applications*, D. Naccache, Ed., vol. 6805 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 383–404. pages 111, 120

[80] Devices, A. M. AMD64 architecture programmer's manual: Volume 2: System programming. AMD Publication no. 24594 rev. 3.11, Dec 2005. pages 2, 20, 97

[81] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13* (Berkeley, CA, USA, 2004), SSYM'04, USENIX Association, pp. 21–21. pages 23

[82] Dobbertin, H., Bosselaers, A., and Preneel, B. RIPEMD-160: A strengthened version of RIPEMD. In *Proceedings of the Third International Workshop on Fast Software Encryption* (London, UK, UK, 1996), Springer-Verlag, pp. 71–82. pages 45, 77

[83] Dodis, Y., Reyzin, L., and Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2004, pp. 523–540. pages 111

[84] Dodis, Y., Reyzin, L., and Smith, A. Fuzzy extractors. In *Security with Noisy Data*, P. Tuyls, B. Skoric, and T. Kevenaar, Eds. Springer London, 2007, pp. 79–99. pages 111

[85] Dolev, D., and Yao, A. C. On the security of public key protocols. Tech. rep., Stanford University, Stanford, CA, USA, 1981. pages 29, 55, 100

[86] Du, W., Deng, J., Han, Y. S., Varshney, P. K., Katz, J., and Khalili, A. A pairwise key predistribution scheme for wireless sensor

networks. *ACM Trans. Inf. Syst. Secur. 8*, 2 (May 2005), 228–258. pages 25, 27

[87] Eckersley, P. How unique is your Web browser? In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies* (Berlin, Heidelberg, 2010), PETS'10, Springer-Verlag, pp. 1–18. pages 23

[88] Edman, M., and Yener, B. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Comput. Surv. 42*, 1 (Dec. 2009), 5:1–5:35. pages 23

[89] Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., and Uhsadel, L. A survey of lightweight-cryptography implementations. *IEEE Design and Test of Computers 24* (2007), 522–533. pages 28, 33

[90] Ekberg, J.-E., and Kylänpää, M. Mobile Trusted Module (MTM) – an introduction. Tech. rep., Nokia Research Center Helsinki, 2007. pages 57, 74, 92

[91] Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., and Sheth, A. N. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2010), OSDI'10, USENIX Association, pp. 1–6. pages 23

[92] Eschenauer, L., and Gligor, V. D. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2002), CCS '02, ACM, pp. 41–47. pages 25, 27

[93] Fiege, U., Fiat, A., and Shamir, A. Zero knowledge proofs of identity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1987), STOC '87, ACM, pp. 210–217. pages 13

[94] Filliâtre, J.-C., and Paskevich, A. Why3 – where programs meet provers. In *Programming Languages and Systems*, M. Felleisen and P. Gardner, Eds., vol. 7792 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 125–128. pages 98, 136

[95] Fishkin, K., and Lundell, J. RFID in healthcare. In *RFID: Applications, Security, and Privacy*, S. Garfinkel and B. Rosenberg, Eds. Addison-Wesley, 2005, pp. 211, 228. pages 47

[96] Ford, W., and Baum, M. S. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption*, 2nd ed. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000. pages 11

[97] Garriss, S., Cáceres, R., Berger, S., Sailer, R., van Doorn, L., and Zhang, X. Trustworthy and personalized computing on public kiosks. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services* (New York, NY, USA, 2008), MobiSys '08, ACM, pp. 199–210. pages 104

[98] Giesecke & Devrient. Mobicore. `http://www.gi-de.com/fra/en/trends_and_insights/mobicore/mobicore_1/mobicore.jsp`. pages 19

[99] Giesecke & Devrient. Sierratee and sierravisor overview. `http://www.sierraware.com/sierraware_tee_hypervisor_overview.pdf`. pages 19

[100] GlobalPlatform. The trusted execution environment: delivering enhanced security at a lower cost to the mobile market. `http://www.globalplatform.org/documents/GlobalPlatform_TEE_White_Paper_Feb2011.pdf`. pages 18

[101] Gobioff, H., Smith, S., Tygar, J. D., and Yee, B. Smart cards in hostile environments. In *Proceedings of the 2Nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2* (Berkeley, CA, USA, 1996), WOEC'96, USENIX Association, pp. 3–3. pages 21, 122

[102] Goldreich, O., Goldwasser, S., and Micali, S. How to construct random functions. *J. ACM 33*, 4 (Aug. 1986), 792–807. pages 9

[103] Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM J. Comput. 18*, 1 (Feb. 1989), 186–208. pages 13

[104] Gollakota, S., Hassanieh, H., Ransford, B., Katabi, D., and Fu, K. They can hear your heartbeats: Non-invasive security for implantable medical devices. *SIGCOMM Comput. Commun. Rev. 41*, 4 (Aug. 2011), 2–13. pages 36

[105] González, J., and Bonnet, P. Towards an open framework leveraging a trusted execution environment. In *Cyberspace Safety and Security*, G. Wang, I. Ray, D. Feng, and M. Rajarajan, Eds., vol. 8300 of *Lecture Notes in Computer Science*. Springer International Publishing, 2013, pp. 458–467. pages 57, 74, 92, 104

[106] Gouvêa, C. P. L., and López, J. High speed implementation of authenticated encryption for the MSP430X microcontroller. In *Proceedings of the 2Nd International Conference on Cryptology and Information Security in Latin America* (Berlin, Heidelberg, 2012), LATINCRYPT'12, Springer-Verlag, pp. 288–304. pages 33

[107] Gürses, F. S. *Multilateral Privacy Requirements Analysis in Online Social Network Services*. PhD thesis, Informatics Section, Department of Computer Science, Faculty of Engineering Science, May 2010. Berendt, Bettina and Preneel, Bart (supervisors). pages 22

[108] He, D., Bu, J., Zhu, S., Chan, S., and Chen, C. Distributed access control with privacy support in wireless sensor networks. *Wireless Communications, IEEE Transactions on 10*, 10 (October 2011), 3472–3481. pages 28

[109] He, D., Bu, J., Zhu, S., Yin, M., Gao, Y., Wang, H., Chan, S., and Chen, C. Distributed privacy-preserving access control in a single-owner multi-user sensor network. In *INFOCOM, 2011 Proceedings IEEE* (April 2011), pp. 331–335. pages 28

[110] Heichlinger, A., and Gallego, P. A new e-ID card and online authentication in Spain. *Identity in the Information Society 3*, 1 (2010), 43–64. pages 15, 123, 127

[111] Hoff, J. V., and Hoff, F. V. The Danish eID case: twenty years of delay. *Identity in the Information Society 3*, 1 (2010), 155–174. pages 10

[112] Hornyack, P., Han, S., Jung, J., Schechter, S., and Wetherall, D. These aren't the droids you're looking for: Retrofitting Android to protect data from imperious applications. In *Proceedings of the 18th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2011), CCS '11, ACM, pp. 639–652. pages 23, 58

[113] Hutter, M., Feldhofer, M., and Plos, T. An ECDSA processor for RFID authentication. In *Workshop on RFID Security - RFIDsec 2010, 6th Workshop, Istanbul, Turkey, June 7-9, 2010, Proceedings* (2010), vol. 6370 of *Lecture Notes in Computer Science*, Springer, pp. 189 – 202. pages 27

[114] Impagliazzo, R., and More, S. M. Anonymous credentials with biometrically-enforced non-transferability. In *Proceedings of the 2003 ACM workshop on Privacy in the electronic society* (New York, NY, USA, 2003), WPES '03, ACM, pp. 60–71. pages 111, 120

[115] Institute for Applied Information Processing and Tu Graz Communications. Crypto software for microcontrollers. http://jce.iaik.tugraz.at/sic/products/crypto_software_for_microcontrollers. pages 27, 33

[116] Institute for Applied Information Processing and Tu Graz Communications. IAIK: JCE-ME. http://jce.iaik.tugraz.at. pages 27

[117] Jacobs, B., Smans, J., and Piessens, F. A quick tour of the Verifast program verifier. In *Proceedings of the 8th Asian Conference on Programming Languages and Systems* (Berlin, Heidelberg, 2010), APLAS'10, Springer-Verlag, pp. 304–311. pages 98, 136

[118] Jain, A. K., Flynn, P., and Ross, A. A. *Handbook of Biometrics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. pages 111

[119] Jain, A. K., Nandakumar, K., and Nagar, A. Biometric template security. *EURASIP J. Adv. Signal Process 2008* (Jan. 2008), 113:1–113:17. pages 111

[120] Jøsang, A., Povey, D., and Ho, A. What you see is not always what you sign. In *Proceedings of the Australian UNIX User Group* (2002), pp. 4–6. pages 21, 95, 122

[121] Juels, A. RFID security and privacy: A research survey. *IEEE J.Sel. A. Commun. 24*, 2 (Sept. 2006), 381–394. pages 26

[122] Juels, A., Molnar, D., and Wagner, D. Security and privacy issues in e-passports. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks* (Washington, DC, USA, 2005), SECURECOMM '05, IEEE Computer Society, pp. 74–88. pages 27

[123] Kauer, B. Oslo: Improving the security of trusted computing. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium* (Berkeley, CA, USA, 2007), SS'07, USENIX Association, pp. 16:1–16:9. pages 98

[124] Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., and Winwood, S. seL4: Formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles* (New York, NY, USA, 2009), SOSP '09, ACM, pp. 207–220. pages 98

[125] Kocher, P. C. On certificate revocation and validation. In *Proceedings of the Second International Conference on Financial Cryptography* (London, UK, UK, 1998), FC '98, Springer-Verlag, pp. 172–177. pages 31

[126] Krawczyk, H., Bellare, M., and Canetti, R. HMAC: Keyed-hashing for message authentication, 1997. pages 45, 77

[127] Krishnamurthy, B., and Wills, C. E. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks* (New York, NY, USA, 2009), WOSN '09, ACM, pp. 7–12. pages 22

[128] Lapon, J., Kohlweiss, M., De Decker, B., and Naessens, V. Analysis of revocation strategies for anonymous Idemix credentials. In *Proceedings of the 12th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security* (Berlin, Heidelberg, 2011), CMS'11, Springer-Verlag, pp. 3–17. pages 5, 12, 73

[129] Law, L., Menezes, A., Qu, M., Solinas, J., and Vanstone, S. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography 28*, 2 (Mar. 2003), 119–134. pages 77

[130] Lee, Y. K., Sakiyama, K., Batina, L., and Verbauwhede, I. Elliptic-curve-based security processor for RFID. *IEEE Trans. Comput. 57*, 11 (Nov. 2008), 1514–1527. pages 27, 28

[131] Lee, Y. K., and Verbauwhede, I. A compact architecture for Montgomery elliptic curve scalar multiplication processor. In *Proceedings of the 8th International Conference on Information Security Applications* (Berlin, Heidelberg, 2007), WISA'07, Springer-Verlag, pp. 115–127. pages 27

[132] Leenes, R., Schallaböck, J., and Hansen, M. Privacy and identity management for Europe. `https://www.prime-project.eu`, May 2008. Last visited on the third of Oktober 2013. pages 17

[133] Leino, K. R. M. Dafny: An automatic program verifier for functional correctness. In *Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning* (Berlin, Heidelberg, 2010), LPAR'10, Springer-Verlag, pp. 348–370. pages 99, 136

[134] Leitold, H., Hollosi, A., and Posch, R. Security architecture of the Austrian citizen card concept. In *ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference* (Washington, DC, USA, 2002), IEEE Computer Society. pages 16

[135] Leroy, X. Formal verification of a realistic compiler. *Commun. ACM 52*, 7 (July 2009), 107–115. pages 99

[136] Liu, D., and Ning, P. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2003), CCS '03, ACM, pp. 52–61. pages 25, 27

[137] Machulak, M. P., Maler, E. L., Catalano, D., and van Moorsel, A. User-managed access to web resources. In *Proceedings of the 6th ACM Workshop on Digital Identity Management* (New York, NY, USA, 2010), DIM '10, ACM, pp. 35–44. pages 14

[138] Mane, D., and Schaumont, P. Energy-architecture tuning for ECC-based RFID tags. In *Radio Frequency Identification*, M. Hutter and J.-M. Schmidt, Eds., Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 147–160. pages 27

[139] McCune, J. M., Li, Y., Qu, N., Zhou, Z., Datta, A., Gligor, V., and Perrig, A. TrustVisor: Efficient TCB reduction and attestation. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2010), SP '10, IEEE Computer Society, pp. 143–158. pages 99

[140] McCune, J. M., Parno, B., Perrig, A., Reiter, M. K., and Seshadri, A. Minimal TCB code execution. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2007), SP '07, IEEE Computer Society, pp. 267–272. pages 95, 98

[141] McCune, J. M., Parno, B., Perrig, A., Reiter, M. K., and Seshadri, A. How low can you go?: recommendations for hardware-supported minimal TCB code execution. *SIGOPS Oper. Syst. Rev. 42*, 2 (Mar. 2008), 14–25. pages 95, 98

[142] McCune, J. M., Parno, B. J., Perrig, A., Reiter, M. K., and Isozaki, H. Flicker: An execution infrastructure for TCB minimization. *SIGOPS Oper. Syst. Rev. 42*, 4 (Apr. 2008), 315–328. pages 2, 95, 98

[143] McCune, J. M., Perrig, A., and Reiter, M. K. Seeing-is-believing: Using camera phones for human-verifiable authentication. *Int. J. Secur. Netw. 4*, 1/2 (Feb. 2009), 43–56. pages 104

[144] Menezes, A. J., Vanstone, S. A., and Oorschot, P. C. V. *Handbook of Applied Cryptography*, 1st ed. CRC Press, Inc., Boca Raton, FL, USA, 1996. pages 8

[145] Miller, S. P., Neuman, B. C., Schiller, J. I., and Saltzer, J. H. Kerberos authentication and authorization system. In *In Project Athena Technical Plan* (1987). pages 34

[146] Mislove, A., Oberoi, G., Post, A., Reis, C., Druschel, P., and Wallach, D. S. AP3: Cooperative, decentralized anonymous communication. In *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop* (New York, NY, USA, 2004), EW 11, ACM. pages 23

[147] Mittal, P., and Borisov, N. Shadowwalker: Peer-to-peer anonymous communication using redundant structured topologies. In *Proceedings of the 16th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2009), CCS '09, ACM, pp. 161–172. pages 23

[148] Molnar, D., and Wagner, D. Privacy and security in library RFID: Issues, practices, and architectures. In *Proceedings of the 11th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2004), CCS '04, ACM, pp. 210–219. pages 27

[149] Morgan, R. L., Cantor, S., Carmody, S., Hoehn, W., and Klingenstein, K. Federated security: The Shibboleth approach. *EDUCAUSE Quarterly 27*, 4 (2004). pages 14

[150] Mostowski, W., and Vullers, P. Efficient U-Prove implementation for anonymous credentials on smart cards. In *Security and Privacy in Communication Networks*, M. Rajarajan, F. Piper, H. Wang, and G. Kesidis, Eds., vol. 96 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer Berlin Heidelberg, 2012, pp. 243–260. pages 13, 111

[151] Nambiar, A., and Wright, M. Salsa: A structured approach to large-scale anonymity. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2006), CCS '06, ACM, pp. 17–26. pages 23

[152] Nauman, M., Khan, S., and Zhang, X. Apex: Extending Android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security* (New York, NY, USA, 2010), ASIACCS '10, ACM, pp. 328–332. pages 23

[153] Naumann, I., and Hogben, G. Privacy features of European eID card specifications. `www.enisa.europa.eu`, 2009. Last visited on the third of October 2013. pages 2, 15

[154] Nicola, M., and John, J. XML parsing: a threat to database performance. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (New York, NY, USA, 2003), CIKM '03, ACM, pp. 175–178. pages 93

[155] Nipkow, T., Wenzel, M., and Paulson, L. C. *Isabelle/HOL: A Proof Assistant for Higher-order Logic*. Springer-Verlag, Berlin, Heidelberg, 2002. pages 98

[156] NIST. Nist sp 800-56a: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. `http://www.nist.gov`. pages 76

[157] Nithyanand, R., Tsudik, G., and Uzun, E. Readers behaving badly. In *Computer Security - ESORICS 2010*, D. Gritzalis, B. Preneel, and M. Theoharidou, Eds., vol. 6345 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 19–36. pages 31

[158] Ohkubo, M., Suzuki, K., and Kinoshita, S. Efficient hash-chain based RFID privacy protection scheme. In *International Workshop on Ubiquitous Computing* (2004). pages 27

[159] Orawiwattanakul, T., Yamaji, K., Nakamura, M., Kataoka, T., and Sonehara, N. User-controlled privacy protection with attribute-filter mechanism for a federated SSO environment using Shibboleth. In *Proceedings of the 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (Washington, DC, USA, 2010), 3PGCIC '10, IEEE Computer Society, pp. 243–249. pages 14

[160] Ottoy, G., Hamelinckx, T., Preneel, B., Strycker, L., and Goemaere, J.-P. AES data encryption in a ZigBee network: Software or hardware? In *Security and Privacy in Mobile Information and Communication Systems*, A. Schmidt, G. Russello, A. Lioy, N. Prasad, and S. Lian, Eds., vol. 47 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer Berlin Heidelberg, 2010, pp. 163–173. pages 29

[161] Parno, B. Bootstrapping trust in a "trusted" platform. In *Proceedings of the 3rd Conference on Hot Topics in Security* (Berkeley, CA, USA, 2008), HOTSEC'08, USENIX Association, pp. 9:1–9:6. pages 104

[162] Partanen, A., and Sievanen, M. FineID specification. Tech. rep., Population Register Centre, 2001. pages 16

[163] Pearson, S. Trusted computing: Strengths, weaknesses and further opportunities for enhancing privacy. In *Trust Management*, P. Herrmann,

V. Issarny, and S. Shiu, Eds., vol. 3477 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 91–117. pages 56

[164] Pearson, S., Mont, M., and Crane, S. Persistent and dynamic trust: Analysis and the related impact of trusted platforms. In *Trust Management*, P. Herrmann, V. Issarny, and S. Shiu, Eds., vol. 3477 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 407–412. pages 56

[165] Pfitzmann, A., and Hansen, M. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, Aug. 2010. v0.34. pages 1, 8

[166] Ågren, M., Hell, M., Johansson, T., and Meier, W. Grain-128a: A new version of Grain-128 with optional authentication. *Int. J. Wire. Mob. Comput. 5*, 1 (Dec. 2011), 48–59. pages 27

[167] Recordon, D., and Reed, D. OpenID 2.0: a platform for user-centric identity management. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management* (New York, NY, USA, 2006), ACM, pp. 11–16. pages 14, 94

[168] Ross, B., Jackson, C., Miyake, N., Boneh, D., and Mitchell, J. C. Stronger password authentication using browser extensions. In *Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14* (Berkeley, CA, USA, 2005), SSYM'05, USENIX Association, pp. 2–2. pages 10

[169] RSA. RSA SecurID. `http://www.emc.com/domains/rsa/index.htm?id=1156`. pages 10

[170] Rushby, J. M. Design and verification of secure systems. *SIGOPS Oper. Syst. Rev. 15*, 5 (Dec. 1981), 12–21. pages 95

[171] Russello, G., Crispo, B., Fernandes, E., and Zhauniarovich, Y. YAASE: Yet another android security extension. In *SocialCom/PASSAT* (2011), IEEE, pp. 1033–1040. pages 23

[172] Sarela, A., Korhonen, I., Lotjonen, J., Sola, M., and Myllymaki, M. IST Vivago - an intelligent social and remote wellness monitoring system for the elderly. In *Information Technology Applications in Biomedicine, 2003. 4th International IEEE EMBS Special Topic Conference on* (April 2003), pp. 362–365. pages 36

[173] Simplício, Jr., M. A., Barreto, P. S. L. M., Margi, C. B., and Carvalho, T. C. M. B. A survey on key management mechanisms for distributed wireless sensor networks. *Comput. Netw. 54*, 15 (Oct. 2010), 2591–2612. pages 25, 27

[174] Song, H., Zhu, S., Zhang, W., and Cao, G. Least privilege and privilege deprivation: Toward tolerating mobile sink compromises in wireless sensor networks. *ACM Trans. Sen. Netw. 4*, 4 (Sept. 2008), 23:1–23:34. pages 28

[175] Spalka, A., Cremers, A. B., and Langweg, H. Trojan horse attacks on software for electronic signatures. *Informatica (Slovenia) 26*, 2 (2002). pages 21, 95, 122

[176] Standaard, D. Recordaantal belgen verloor vorig jaar identiteitskaart. `http://www.standaard.be/cnt/dmf20100414_002`. pages 71

[177] Stein, M. Mobile devices as secure eID reader using trusted execution environments. Presented at Open Identity Summit, 2013. pages 57, 74, 92, 104

[178] Sterckx, M., Gierlichs, B., Preneel, B., and Verbauwhede, I. Efficient implementation of anonymous credentials on Java Card smart cards. In *1st IEEE International Workshop on Information Forensics and Security (WIFS 2009)* (London,UK, 2009), IEEE, pp. 106–110. pages 13, 111

[179] Strackx, R., Noorman, J., Verbauwhede, I., Preneel, B., and Piessens, F. Protected software module architectures. In *ISSE 2013 Securing Electronic Business Processes*, H. Reimer, N. Pohlmann, and W. Schneider, Eds. Springer Fachmedien Wiesbaden, 2013, pp. 241–251. pages 99, 135

[180] Strackx, R., and Piessens, F. Fides: Selectively hardening software application components against kernel-level or process-level malware. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (New York, NY, USA, 2012), CCS '12, ACM, pp. 2–13. pages 99, 135

[181] Sun, L. Security and privacy on low-cost radio frequency identification systems. *Int. J. Secur. Netw. 5*, 2/3 (Mar. 2010), 128–134. pages 27

[182] Suriadi, S., Foo, E., and Jøsang, A. A user-centric federated single sign-on system. *J. Netw. Comput. Appl. 32*, 2 (Mar. 2009), 388–401. pages 17

[183] Syverson, P., Tsudik, G., Reed, M., and Landwehr, C. Towards an analysis of onion routing security. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability* (New York, NY, USA, 2001), Springer-Verlag New York, Inc., pp. 96–114. pages 23

[184] Tabar, A. M., Keshavarz, A., and Aghajan, H. Smart home care network using sensor fusion and distributed vision-based reasoning. In *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks* (New York, NY, USA, 2006), VSSN '06, ACM, pp. 145–154. pages 36

[185] Trusted Computing Group. TCG TNC. `https://www.trustedcomputinggroup.org/developers/trusted_network_connect`. pages 119

[186] Trusted Computing Group. TCG TPM specification. `http://www.trustedcomputinggroup.org/resources/tpm_main_specification`. pages 2, 20, 96, 100

[187] Tsyrklevich, E., and Tsyrklevich, V. Single sign-on for the Internet: A security story. `https://www.blackhat.com`, 2007. pages 15

[188] Van Rijswijk, R. M., and Van Dijk, J. Tiqr: A novel take on two-factor authentication. In *Proceedings of the 25th International Conference on Large Installation System Administration* (Berkeley, CA, USA, 2011), LISA'11, USENIX Association, pp. 7–7. pages 10

[189] van Rijswijk-Deij, R., and Poll, E. Using trusted execution environments in two-factor authentication: comparing approaches. In *Proceedings Open Identity Summit 2013. Open Identity Summit (OID-2013), September 9-11, Kloster Banz, Germany* (2013), vol. 223 of *Lecture Notes in Informatics, LNI*, Gesellschaft für Informatik, Springer, pp. 20–31. pages 57, 74, 92

[190] Vasudevan, A., Chaki, S., Jia, L., McCune, J., Newsome, J., and Datta, A. Design, implementation and verification of an extensible and modular hypervisor framework. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2013), SP '13, IEEE Computer Society, pp. 430–444. pages 99, 135

[191] Vasudevan, A., Owusu, E., Zhou, Z., Newsome, J., and McCune, J. M. Trustworthy execution on mobile devices: What security properties can my mobile platform give me? In *Proceedings of the 5th International Conference on Trust and Trustworthy Computing* (Berlin, Heidelberg,

2012), TRUST'12, Springer-Verlag, pp. 159–178. pages 19, 57, 74, 92, 104

[192] Vasudevan, A., Parno, B., Qu, N., Gligor, V. D., and Perrig, A. Lockdown: Towards a safe and practical architecture for security applications on commodity platforms. In *Proceedings of the 5th International Conference on Trust and Trustworthy Computing* (Berlin, Heidelberg, 2012), TRUST'12, Springer-Verlag, pp. 34–54. pages 105

[193] Verhaeghe, P., Lapon, J., Decker, B., Naessens, V., and Verslype, K. Security and privacy improvements for the Belgian eID technology. In *Emerging Challenges for Security, Privacy and Trust*, D. Gritzalis and J. Lopez, Eds., vol. 297 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2009, pp. 237–247. pages 11

[194] Vossaert, J., Lapon, J., De Decker, B., and Naessens, V. Symmetric key infrastructure for authenticated key establishment between resource constrained nodes and powerful devices. *Security and Communication Networks* (2011), n/a–n/a. pages 4

[195] Vossaert, J., Lapon, J., De Decker, B., and Naessens, V. User-centric identity management using trusted modules. In *Proceedings of the 7th European Conference on Public Key Infrastructures, Services and Applications* (Berlin, Heidelberg, 2011), EuroPKI'10, Springer-Verlag, pp. 155–170. pages 5

[196] Vossaert, J., Lapon, J., De Decker, B., and Naessens, V. User-centric identity management using trusted modules. In *Proceedings of the 7th European Conference on Public Key Infrastructures, Services and Applications* (Berlin, Heidelberg, 2011), EuroPKI'10, Springer-Verlag, pp. 155–170. pages 5

[197] Vossaert, J., Lapon, J., De Decker, B., and Naessens, V. Trusted computing to increase security and privacy in eID authentication. In *ICT Systems Security and Privacy Protection*, N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. Abou El Kalam, and T. Sans, Eds., vol. 428 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2014, pp. 485–492. pages 6

[198] Vossaert, J., Lapon, J., Decker, B., and Naessens, V. Personalized mobile services with lightweight security in a sports association. In *Security and Privacy in Mobile Information and Communication Systems*, A. U. Schmidt, G. Russello, A. Lioy, N. Prasad, and S. Lian, Eds., vol. 47 of *Lecture Notes of the Institute*

for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2010, pp. 3–14. pages 4

[199] Vossaert, J., Lapon, J., Decker, B., and Naessens, V. Client-side biometric verification based on trusted computing. In *Communications and Multimedia Security*, B. Decker, J. Dittmann, C. Kraetzer, and C. Vielhauer, Eds., vol. 8099 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 34–49. pages 6

[200] Vossaert, J., Lapon, J., Decker, B. D., and Naessens, V. User-centric identity management using trusted modules. *Mathematical and Computer Modelling 57*, 7-8 (2013), 1592 – 1605. pages 5

[201] Vullers, P., and Alpár, G. Efficient selective disclosure on smart cards using Idemix. In *Policies and Research in Identity Management*, S. Fischer, E. Leeuw, and C. Mitchell, Eds., vol. 396 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2013, pp. 53–67. pages 13, 111, 120

[202] Wang, H., and Li, Q. Distributed user access control in sensor networks. In *Proceedings of the Second IEEE International Conference on Distributed Computing in Sensor Systems* (Berlin, Heidelberg, 2006), DCOSS'06, Springer-Verlag, pp. 305–320. pages 28

[203] Wang, R., Chen, S., and Wang, X. Signing me onto your accounts through Facebook and Google: A traffic-guided security study of commercially deployed single-sign-on web services. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2012), SP '12, IEEE Computer Society, pp. 365–379. pages 15

[204] Wenger, E., and Werner, M. Evaluating 16-bit processors for elliptic curve cryptography. In *Proceedings of the 10th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications* (Berlin, Heidelberg, 2011), CARDIS'11, Springer-Verlag, pp. 166–181. pages 27, 28

[205] Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., and Memon, N. Passpoints: Design and longitudinal evaluation of a graphical password system. *Int. J. Hum.-Comput. Stud. 63*, 1-2 (July 2005), 102–127. pages 10

[206] Winter, J. Experimenting with ARM TrustZone – or: How I met friendly piece of trusted hardware. In *Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications* (Washington, DC, USA, 2012), TRUSTCOM '12, IEEE Computer Society, pp. 1161–1166. pages 57, 74, 92, 104

[207] Wojtczuk, R., and Rutkowska, J. Attacking Intel trusted execution technology, 2009. pages 105

[208] Wojtczuk, R., Rutkowska, J., and Tereshkin, A. Another way to circumvent Intel trusted execution technology, 2009. pages 105

[209] Yee, K.-P. User interaction design for secure systems. In *Proceedings of the 4th International Conference on Information and Communications Security* (London, UK, UK, 2002), ICICS '02, Springer-Verlag, pp. 278–290. pages 20

[210] Yu, S., Ren, K., and Lou, W. FDAC: Toward fine-grained distributed data access control in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on 22*, 4 (April 2011), 673–686. pages 28

[211] Zhang, M., and Fang, Y. Security analysis and enhancements of 3GPP authentication and key agreement protocol. *Trans. Wireless. Comm. 4*, 2 (Mar. 2005), 734–742. pages 34

[212] Zhou, Z., Gligor, V. D., Newsome, J., and McCune, J. M. Building verifiable trusted path on commodity x86 computers. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2012), SP '12, IEEE Computer Society, pp. 616–630. pages 99

List of Publications

Main Publications

Articles in internationally reviewed academic journals

1. Jan Vossaert, Jorn Lapon, Bart De Decker, Vincent Naessens. User-centric identity management using trusted modules. *Mathematical and Computer Modelling*, 57 (7-8), 1592-1605, 2013

2. Jan Vossaert, Jorn Lapon, Bart De Decker, Vincent Naessens. Symmetric key infrastructure for authenticated key establishment between resource constrained nodes and powerful devices. *Security and Communication Networks*, 2011

Papers at international scientific conferences and symposia, published in full in proceedings

3. Jan Vossaert, Jorn Lapon, Bart De Decker, Vincent Naessens. Trusted computing to increase security and privacy in eID authentication. *IFIP SEC International Information Security and Privacy Conference.* Marrakech, Morocco, 2-4 June 2014

4. Jan Vossaert, Jorn Lapon, Vincent Naessens. Out-of-band password based authentication towards web services. In Lieven De Strycker, editor, *ECUMICT 2014*, volume 302 of *Proceedings Lecture Notes in Electrical Engineering*, pages 181-191, Springer International Publishing, Ghent, 27-28 March 2014

5. Jan Vossaert, Jorn Lapon, Bart De Decker, Vincent Naessens. Client-side biometric verification based on trusted computing. In Bart De Decker, Jana Dittmann, Christian Kraetzer, Claus Vielhauer, editors,

Communications and Multimedia Security, Volume 8099 of *Lecture Notes in Computer Science*, pages 34-49, Berlin Heidelberg: Springer, Magdeburg, Germany, 25-26 September 2013

6. Jan Vossaert, Jorn Lapon, Bart De Decker, Vincent Naessens. User-centric identity management using trusted modules. In Jan Camenisch, Costas Lambrinoudakis, editors, *Public Key Infrastructures, Services and Applications*, Volume 6711 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pages 155-170, Athens, Greece, 23-24 September 2010

7. Jan Vossaert, Pieter Verhaeghe, Bart De Decker, Vincent Naessens. A smart card based solution for user-centric identity management. In Simone Fischer-Hübner, Penny Duquenoy, Marit Hansen, Ronald Leenes, Ge Zhang, editors, *Privacy and Identity Management for Life*, Volume 352 of *IFIP Advances in Information and Communication Technology*, Springer Berlin Heidelberg, pages 164-177, Helsingborg, Sweden, 2-6 Aug 2010

8. Jan Vossaert, Jorn Lapon, Bart De Decker, Vincent Naessens. Personalized mobile services with lightweight security in a sports association. In Andreas U. Schmidt, Giovanni Russello, Antonio Lioy, Neeli R. Prasad, Shinguo Lian, editors, *Security and Privacy in Mobile Information and Communication Systems*, Volume 47 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer Berlin Heidelberg, pages 3-14, Catania, 27-28 May 2010

Meeting abstracts, presented at international scientific conferences and symposia

9. Jan Vossaert, Jorn Lapon, Vincent Naessens. Towards a cross-domain identity card. 4th Benelux Workshop on Information and System Security, Louvain-La-Neuve, 19-20 November 2009

Internal reports

10. Jan Vossaert, Pieter Verhaeghe, Bart De Decker, Vincent Naessens. Towards a general purpose identity card. CW Reports CW587, Department of Computer Science, K.U.Leuven, 2010

Publications with Contributions as Co-author

Papers at international scientific conferences and symposia, published in full in proceedings

11. Faysal Boukayoua, Jan Vossaert, Bart De Decker, Vincent Naessens. Classification of advanced mobile access control scenarios. In Lieven De Strycker, editor, *Proceedings of the Fifth European Conference on the Use of Modern Information and Communication Technologies*, Nevelland Graphics cvba-so, pages 175-186, Ghent, 22-23 March 2012

12. Faysal Boukayoua, Jan Vossaert, Bart De Decker, Vincent Naessens. Claim-based versus network-based identity management: a hybrid approach, In Andreas U. Schmidt, Giovanni Russello, Ioannis Krontiris, Shinguo Lian, editors, *Security and Privacy in Mobile Information and Communication Systems*, Volume 107 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer Berlin Heidelberg, pages 38-50, Frankfurt am Main, 25-26 June 2012

13. Faysal Boukayoua, Jan Vossaert, Bart De Decker, Vincent Naessens. Using a smartphone to access personalized Web services on a workstation, In Jan Camenisch, Bruno Crispo, Simone Fischer-Hübner, Ronald Leenes, Giovanni Russello, editors, *Privacy and Identity Management for Life*, Volume 375 of *IFIP Advances in Information and Communication Technology*, Springer Berlin Heidelberg, pages 144-156, Trento, Italy, 5-9 September 2011

Meeting abstracts, presented at international scientific conferences and symposia

14. Koen Decroix, Jan Vossaert, Bart De Decker, Vincent Naessens. Reconfigurable security in home monitoring systems. 5th Benelux Workshop on Information and System Security, Nijmegen, the Netherlands, 29-30 Nov 2010

Internal reports

15. Faysal Boukayoua, Italo Dacosta, Bart De Decker, Jorn Lapon, Luc Martens, Milica Milutinovic, Vincent Naessens, Bart Preneel, Andreas Put, Stefaan Seys, Dave Singelee, Kris Vanhecke, Jan Vossaert. MobCom

Deliverable: D.3.1 Development of Architecture; D.4.1 Applications I; D.5.1 Validation Requirements, 2012

16. Faysal Boukayoua, Bart De Decker, Tom De Ryckere, Luc Martens, Milica Milutinovic, Vincent Naessens, Bart Preneel, Stefaan Seys, Dave Singelee, Kris Vanhecke, Jan Vossaert. MobCom Deliverable: D.2.1 Interim Report on Basic Research, 2012

17. Faysal Boukayoua, Bart De Decker, Luc Martens, Vincent Naessens, Bart Preneel, Stefaan Seys, Kris Verslype, Jan Vossaert. MobCom Deliverable: D.1.1 Report on state-of-the-art and requirements study I, 2011

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
IMINDS-DISTRINET
Celestijnenlaan 200A box 2402
B-3001 Heverlee
jan.vossaert@cs.kuleuven.be
http://wms.cs.kuleuven.be/cs/